FADER: Fast Adversarial Example Rejection

Francesco Crecchi[a,1], Marco Melis[b,1], Angelo Sotgiu[b,1], Davide Bacciu[a], Battista Biggio[b]

[a]*Dipartimento di Informatica, Università di Pisa, Italy.*
[b]*Dipartimento di Ingegneria Elettrica ed Elettronica, Università di Cagliari, Italy.*

Abstract

Deep neural networks are vulnerable to adversarial examples, i.e., carefully-crafted inputs that mislead classification at test time. Recent defenses have been shown to improve adversarial robustness by detecting anomalous deviations from legitimate training samples at different layer representations - a behavior normally exhibited by adversarial attacks. Despite technical differences, all aforementioned methods share a common backbone structure that we formalize and highlight in this contribution, as it can help in identifying promising research directions and drawbacks of existing methods. The first main contribution of this work is the review of these detection methods in the form of a unifying framework designed to accommodate both existing defenses and newer ones to come. In terms of drawbacks, the overmentioned defenses require comparing input samples against an oversized number of reference prototypes, possibly at different representation layers, dramatically worsening the test-time efficiency. Besides, such defenses are typically based on ensembling classifiers with heuristic methods, rather than optimizing the whole architecture in an end-to-end manner to better perform detection. As a second main contribution of this work, we introduce FADER, a novel technique for speeding up detection-based methods. FADER overcome the issues above by employing RBF networks as detectors: by fixing the number of required prototypes, the runtime complexity of adversarial examples detectors can be controlled. Our experiments outline up to 73× prototypes reduction compared to analyzed detectors for MNIST dataset, up to 50× for CIFAR10 dataset, and up to 82× on ImageNet10 dataset respectively, without sacrificing classification accuracy on both clean and adversarial data.

Keywords: adversarial machine learning; adversarial examples; detection; evasion attacks; rbf networks; deep learning

1. Introduction

In recent years, Deep Neural Networks (DNNs) achieved state-of-the-art performances in a wide variety of pattern recognition tasks, including (but not limited to) image classification [1], natural language processing [2] and reinforcement learning [3]. These impressive results, made DNNs appealing for building *smart applications*, i.e. software embedding an *intelligent* component in a form of a DNN to perform pattern recognition, planning, and recommendations. Today, deep learning is a core component for software spanning from consumer to safety-critical applications like autonomous driving [4], malware and intrusion detection [5], homeland security [6], and medical diagnosis [7].

Deep learning (DL) models, as other machine learning algorithms, are designed to work under the so-called *stationarity assumption*: the training data distribution and that of the test samples are assumed to be *the same*. However, distribution drifts can happen *naturally*, e.g., missing data due to sensor failure, or *adversarially*, i.e., an adversary that tampers with data purposely to cause failures during system operation [8, 9]. While DNNs are known to be robust to random noise, it has been shown that the accuracy of DNNs and, in general, of machine-learning algorithms, can dramatically deteriorate in face of gradient-based adversarial attacks [10, 11], including adversarial examples, i.e., carefully-perturbed input samples that mislead classification at test time [12, 13]. A plethora of methods have been proposed to find adversarial examples [14, 15, 16, 17, 18, 19, 20, 21]. These often transfer across different architectures, enabling black-box attacks even for inaccessible models [22]. The vulnerability of DL models to adversarial samples has the potential to make them the weakest link in the security chain of smart applications. A great number of countermeasures to adversarial examples have been deployed during recent years, still leaving this as an open research problem.

We can categorize defense mechanisms against adversarial examples into two main complementary groups: *robust learning* and *detection* methods [9]. The former typically employ *adversarial training* [13], i.e., retrain the model on adversarial examples to improve classifier robustness against specific attack algorithms. This requires, however, generating attack samples during model training, which may be very computationally demanding for state-of-the-art DNNs.

Detection approaches, instead, include explicit detection or rejection strategies for adversarial samples, i.e., they provide an additional class for anomalies and potential out-of-distribution attacks. Typically, these defenses are designed to work under the so-called *manifold hypothesis*: in several domains, natural

Email addresses: francesco.crecchi@di.unipi.it (Francesco Crecchi), marco.melis@unica.it (Marco Melis), angelo.sotgiu@unica.it (Angelo Sotgiu), bacciu@di.unipi.it (Davide Bacciu), battista.biggio@unica.it (Battista Biggio)

[1]Equal contribution.

data are assumed to lie in a low-dimensional manifold embedded in a high-dimensional space (e.g., grayscale digits image domain). Remarkably, not every high-dimensional representation belongs to the natural data manifold (e.g., salt and pepper noise). Assuming adversarial examples to be out-of-manifold data, manifold-based defenses work by identifying adversarial points from their distance to the manifold and, optionally, by "pulling them back" onto the data manifold before classification. These defenses are based on a distance-based rejection strategy: as far as a sample moves away from class prototypes, classifier support decreases to zero. If an input sample is not supported by any class, then it is rejected. Remarkable instances of this approach in the literature are found in [14, 15, 16, 17, 20, 23, 19, 18]. Apart from technical differences, all these rejection-based defenses share a common backbone structure which can be abstracted as a framework.

The first main contribution of this work is to provide a comprehensive review of such adversarial examples detection methods in the form of a unifying framework. Each proposed detector defense can be obtained by correctly instantiating our framework components. Subsuming each analyzed detector defense in the framework allowed us to identify common drawbacks, leading us to the second main contribution of this paper.

The vast majority of detector defenses in literature are a form of instance-based classifiers: when a new sample is fed to the classifier, it is compared with a set of prototypes to produce an output prediction. The number of selected training prototypes is thus crucial for the runtime efficiency of the detector, which we found to be not properly tuned in literature solutions. A second issue we found in existing approaches is that, for multilayered defenses, classifiers are optimized to maximize detection separately, i.e. they are not jointly trained to perform rejection. They are typically based on ensembling classifiers with heuristic methods, rather than optimizing the whole architecture in an end-to-end manner to better perform detection.

To overcome these limitations, in this paper, we propose FADER, a technique for speeding up detection methods. It works by replacing the detector's distance-based classifiers with size-constrained RBF networks, to reduce computational overhead at test time. The proposed solution is capable of enforcing adversarial robustness even in presence of adaptive attacks specifically designed to defeat such defense (see Section 4).

In summary, we make the following contributions:

- Comprehensive literature review of the detector-based defenses to provide a unified view in the form of a framework, which helps identify current defenses limitations.
- Overcoming such limitations by proposing FADER, i.e., a technique to obtaining an end-to-end differentiable detector capable of an up to 80× prototypes reduction with respect to analyzed competitors.
- Novel adaptive attack algorithm designed for the proposed defense method to avoid security by gradient obfuscation.

The rest of the paper is structured as follows. In Section 2 we present our adversarial examples detector framework. Section 3 introduces FADER, our proposed fast detection method. Section 4 devises how to perform a reliable security evaluation properly and the adaptive attack specifically designed to evaluate our proposed fast detector. FADER-based solutions are empirically evaluated in Section 5, testing adversarial attack detection in different image recognition tasks. Section 6 discusses complementary methods for addressing adversarial examples than detection strategies. We conclude the paper by discussing the main contributions of this work and its limitations, along with promising future research directions (Section 7).

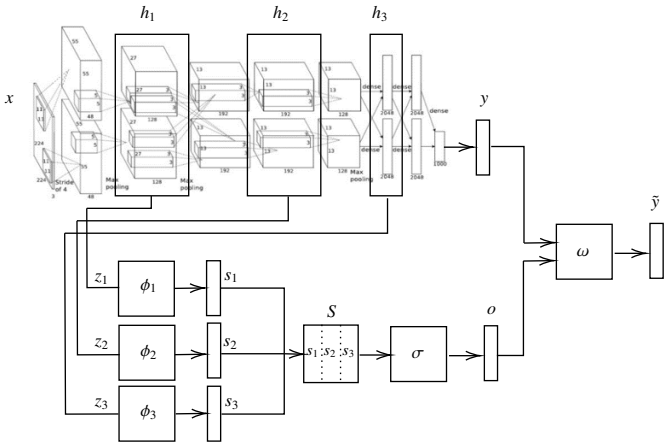

Figure 1: Architecture of the proposed framework for adversarial example detection. It extends a pre-trained DNN by attaching several layer detectors ϕ whose goal is to determine distribution drifts in the representation of an input $\boldsymbol{x}$ at a given layer. Multiple layer-detector predictions are combined and fed to a combiner classifier σ which outputs the final detector prediction. The undefended network and detector outputs are combined by ω to provide the final predictions.

2. A Framework for Adversarial Example Detection

The proposed detection framework, conceptually depicted in Fig. 1, assumes an already-trained DNN classifier to be protected against adversarial examples, denoted with $f : \mathcal{X} \to \mathcal{Y}$, being $\mathcal{X} \subseteq \mathbb{R}^d$ a d-dimensional input space, and $\mathcal{Y} \subseteq \mathbb{R}^c$ the output space consisting of c classes.[2] Let us assume that, without loss of generality, the network consists of m layers. This implies that the prediction function f can be expressed as a composition of functions $f(h_m(h_{m-1}(\dots h_1(\boldsymbol{x}; \boldsymbol{w}_1)); \boldsymbol{w}_{m-1}); \boldsymbol{w}_m)$, where h_1 and h_m denote the mapping function learned, respectively, by the input and the output layer, and $\boldsymbol{w}_1$ and $\boldsymbol{w}_m$ are their weight parameters (learned during training).

Building a detector amounts to selecting a set of network layers, to inspect the internal DNN representations corresponding to each given input sample, with the goal of identifying anomalous (adversarial) patterns with respect to those exhibited by the natural samples. Let $\boldsymbol{z}_j = h_j(h_{j-1}(\dots(h_1(\boldsymbol{x}; \boldsymbol{w}_1)); \boldsymbol{w}_{j-1}); \boldsymbol{w}_j)$ be the representation learned by the network for a given input sample $\boldsymbol{x}$ at layer h_j. A *layer detector* function $\phi_j : \mathcal{Z}_j \to \mathcal{S}_j$

[2] We assume here that the classifier predicts continuous output values for each class, and that the final decision is made as usual by selecting the class exhibiting the maximum support.

is applied to z_j producing a score vector s_j of size c, where each element represents the probability that x belongs to a given class, according to the given representation z_j.

The predictions coming from the k different layer detectors are then stacked in a scoring matrix $\mathbf{S} \in \mathbb{R}^{k \cdot c}$, and combined by a *multilayer detector* $\sigma : \mathbb{R}^{k \cdot c} \rightarrow O$, where $o \in O$ is the vector of predictions provided by the multilayer detector. The multilayer detector may provide one output for each class, i.e., $O = \mathcal{Y}$, or alternatively only produce a single score that measures how likely the input sample is retained anomalous.

The predictions of the multilayer detector may be finally combined with those coming from the undefended DNN to produce the final predictions for each input sample x. This can be formalized as a function $\omega : \mathcal{Y} \times O \rightarrow \tilde{\mathcal{Y}}$, being $y \in \mathcal{Y}$ and $o \in O$ the DNN and the detector predictions, respectively, and $\tilde{y} \in \tilde{\mathcal{Y}}$ a $c + 1$ output vector which includes an additional rejection class reserved for the detected adversarial examples. For single-layer defenses, no combiner is clearly needed. In our framework this corresponds to instantiating σ as the identity function.

In the remainder of the section, we will rephrase the aforementioned adversarial-example defenses in terms of the proposed framework. As already mentioned, we are considering only rejection-based defenses against adversarial attacks. For the sake of clarity, the considered defenses are also schematized in terms of framework components in Table 1.

2.1. Neural Reject

Inspired by the concept of *open set* recognition, Melis et al. [15] proposed a method called Neural Reject (NR), which attaches a Support Vector Machine with an RBF kernel (SVM RBF) on the last hidden layer of a DNN to perform rejection of samples showing an outlying behavior. In particular, the choice of the RBF kernel implies that the prediction scores provided by the SVM are proportional to the distance of the input sample to the reference prototypes (i.e., the support vectors), thus enabling rejection of samples which fall far away from the training data in the given representation space. This single-layered defense can be expressed in our framework, instantiating ϕ_{m-1} as an SVM RBF.

2.2. Kernel Density Estimation

Feinman et al. [14] proposed an adversarial examples detector exploiting a Kernel Density Estimator (KDE) on the embeddings obtained from the last hidden layer of the neural network to identify low confidence input regions. As for NR, such defense can be obtained by instantiating ϕ_{m-1} as a KDE.

2.3. DNN Binary Classifier

The idea of a layer-wise detector is further developed in [23] providing a single detector subnetwork connected to an arbitrary layer of the DNN which is intended to protect. This subnetwork is trained to perform a binary classification task to distinguish genuine data from samples containing adversarial perturbations. In our framework, ϕ_j is the detector subnetwork at a given layer j.

2.4. Dimensionality Reduction

Multiple-layers inspection has been performed by Crecchi et al. [19], who proposed a detection method combining nonlinear dimensionality reduction techniques (i.e. t-SNE [24]) and density estimation to detect adversarial samples. For a given layer j of the network, the classifier obtained by performing density estimation on top of the embeddings produced by t-SNE represents ϕ_j, whereas the support vector machine combiner is a realization of σ.

2.5. Deep Neural Reject

Sotgiu et al. [16] proposed to apply NR to multiple internal layer representations to form a Deep Neural Rejection (DNR) detector, empirically demonstrating improvements upon the single-layered solution. As for NR, ϕ_j is obtained through SVM RBF classifiers at layer j, whereas σ is again an SVM RBF, trained via *stacked generalization* [25].

2.6. Deep k-Nearest Neighbour

Papernot & McDaniel proposed a detection method named Deep k-Nearest Neighbour (DkNN) [17], which employs a k-nearest neighbor classifier on the representations of the data learned by each layer of the DNN. When a test input is fed to DkNN, it is compared to its neighboring training points according to the distance that separates them in the representations to estimate the nonconformity, i.e. the lack of support, for a prediction in the training data. If the input sample is not conformed with the training data, it is rejected as an adversarial example. This defense can be obtained by employing kNNs for ϕ_j for layer j of the DNN. Statistical hypothesis test for combiner predictions in the realm of *conformal predictions* [26, 27, 28] can be used as σ.

2.7. Generative Models

As generative models are trained to *approximate* the data generating distribution (which is typically unknown), they are a natural candidate for manifold-based defenses against adversarial examples. Meng & Cheng proposed MagNet [18] for defending neural network classifiers against adversarial samples leveraging generative models. MagNet works in the input space and employs one or more separate detector networks in the form of a denoising autoencoder (DAE) exploiting the reconstruction error to estimate how far a test sample is from the manifold of normal samples and to *reform* it to a natural sample lying on the data manifold, which is used for classification.

Fortified Networks [20] exploit this very same idea but on the learned hidden representation distribution: DAEs are inserted at crucial points between layers of the original DNN to *clean-up* the adversarial sample lying away from the original data manifold, arguing that this provides stronger protection against adversarial examples than acting in the input space.

Magnet and Fortified Network defenses can be obtained in our framework by instantiating ϕ_j as a DAE, for layer j. By having σ as the identity function, threshold-based detection (ω) on input sample reconstruction error can be used to identity outliers to the expected input distribution.

Defense	Adv. Training	ϕ	σ
Feinman et al. [14]	✗	KDE	-
Melis et al. [15]	✗	SVM RBF	-
Sotgiu et al. [16]	✗	SVM RBF	SVM RBF
Papernot et al. [17]	✗	k-NN	Statistical Test
Lamb et al. [20]	✗	DAE	-
Metzen et al. [23]	✓	DNN	-
Crecchi et al. [19]	✓	t-SNE + KDE	SVM
Meng & Cheng [18]	✓	DAE	-

Table 1: Detector-based defenses against adversarial examples framed in our proposed detector framework (− for unnecessary components).

However, despite the promising theoretical background, all these methods are still vulnerable [29, 30].

3. Fast Adversarial Example Rejection

In this section, we present our proposal for speeding up existing detection methods for adversarial examples by controlling the number of reference prototypes they make use of. Previous instance-based detectors [15, 16, 14, 17, 19], in fact, do not allow one to specify the number of prototypes (e.g. *support vectors* for SVM-based ones) used for identifying adversarial examples. Selecting a large number of reference prototypes, possibly at different representation layers, dramatically increases classification time, as the input sample has to be compared with each prototype at each selected representation layer to compute the corresponding prediction. Thus, controlling the number of prototypes employed by detectors is crucial for runtime efficiency. With FADER, we propose to replace existing classifiers in such detectors with size-constrained RBF networks designed for an optimal accuracy vs. speed trade-off.

RBF networks are *shallow* artificial neural networks that use radial basis functions (RBF) as activation functions. The output of the network is a linear combination of radial basis functions of the inputs and neuron parameters. Despite their architectural simplicity, they have been shown to possess structural resistance to adversarial attacks [13, 31, 32, 33] thanks to their localized nature, thus they are a natural candidate for building fast and secure detectors. The use of RBF activation functions enforces the classifier to assume a desirable *compact abating probability* property for open set recognition [34, 35]. Being $s_1, \ldots, s_c$ the output scores produced by the classifier for an input sample $\boldsymbol{x}$, such property ensures that, for each given class, such scores decrease while $\boldsymbol{x}$ moves away from input regions densely populated by training samples of that class. This property allows us to easily implement a *distance-based rejection* mechanism required to detect adversarial examples, as also suggested in [36, 34, 16].

3.1. FADER

The central idea of FADER is to speed up instance-based adversarial example detectors by controlling the number of prototypes used for comparison while maintaining comparable performances with original solutions. To this end, we replace detector classifiers with RBF network-based ones, allowing for the joint optimization of prototypes and network parameters. Suppose to take an RBF SVM as a reference layer detector (e.g., used in NR or DNR) that we intend to speed up. Then, given $\boldsymbol{z}_j$ as the input representation for an input sample $\boldsymbol{x}$ obtained at the j-th layer of the DNN, the layer-detector function can be formulated as follows:

$$\phi_j(\boldsymbol{z}_j) = \text{sign}\left(\sum_{i=1}^{n} \alpha_{j,i} \exp\left(-\gamma_j \|\boldsymbol{z}_j - \boldsymbol{z}_{j,i}\|^2\right) + b_j\right), \quad (1)$$

where $\boldsymbol{z}_{j,i}$ is the representation of the training sample $\boldsymbol{x}_i$ at layer j, $\alpha_{j,i}$ is its corresponding (signed) coefficient, b_j is the bias, and γ_j is the RBF kernel parameter. The coefficients $\alpha_{j,i}$ and b are learned via SVM training. One issue with kernelized SVMs is that the number of support vectors (i.e., the training samples for which $\alpha_{j,i} \neq 0$) grows linearly with the training set size [37, 38, 39], and it tends to match the training set size n when considering multiclass SVMs (as the union of the support vector sets learned in a disjoint manner by the different binary SVMs tends to be non-sparse). The runtime complexity of RBF SVMs is $O(p \cdot n_{\text{sv}} \cdot d)$ [40], i.e., it scales linearly with the number of test samples p, the number of support vectors n_{sv}, and the dimensionality d of the input (at the considered layer). Accordingly, RBF SVMs tend to become too computationally demanding at runtime when trained on large training sets. In addition, they also require storing a much larger number of reference prototypes in memory, which hinders portability on low-memory embedded devices. These problems are also witnessed by the fact that a substantial amount of previous work has proposed many different SVM variants aimed to reduce or prune the set of support vectors to speed up classification and reduce memory consumption [40, 39].

FADER aims to replace the RBF SVM layer detectors with an RBF network, to reduce the number of prototypes while maintaining the desired detector behavior, i.e., nearly the *same* decision regions of the unpruned SVMs (see Fig. 2). The new detector decision function can be formulated as follows:

$$\phi_j(\boldsymbol{z}_j) = \text{sign}\left(\sum_{i=1}^{n'} \beta_{j,i} \exp\left(-\gamma_{j,i} \|\boldsymbol{z}_j - \boldsymbol{\mu}_{j,i}\|^2\right) + b_j\right). \quad (2)$$

Although the two definitions look alike, they substantially differ in practical terms. The reference prototypes $\boldsymbol{\mu}_{j,i}$ can now be optimized during RBF network training, as well as the kernel parameters $\gamma_{j,i}$ (one per prototype), to better fit the training data. This improved flexibility allows us to drastically reduce the number of reference prototypes. Moreover, jointly optimizing prototypes ($\boldsymbol{\mu}_{j,i}$), kernel ($\gamma_{j,i}$) and network parameters ($\beta_{j,i}, b_j$) enables a significant reduction of the number of prototypes, while maintaining comparable performances with respect to over-specified solutions, as shown in our experiments (see Section 5). In addition, n' does not need to scale linearly with the training set size (as it is fixed a priori), thereby significantly reducing runtime complexity and memory consumption.

In terms of the proposed adversarial detector framework in Section 2, FADER can be represented as follows: the layer detector function ϕ can be instantiated as an RBF network. In the case of multilayered detectors, the combiner σ can be instantiated again as an RBF network. Adversarial examples are

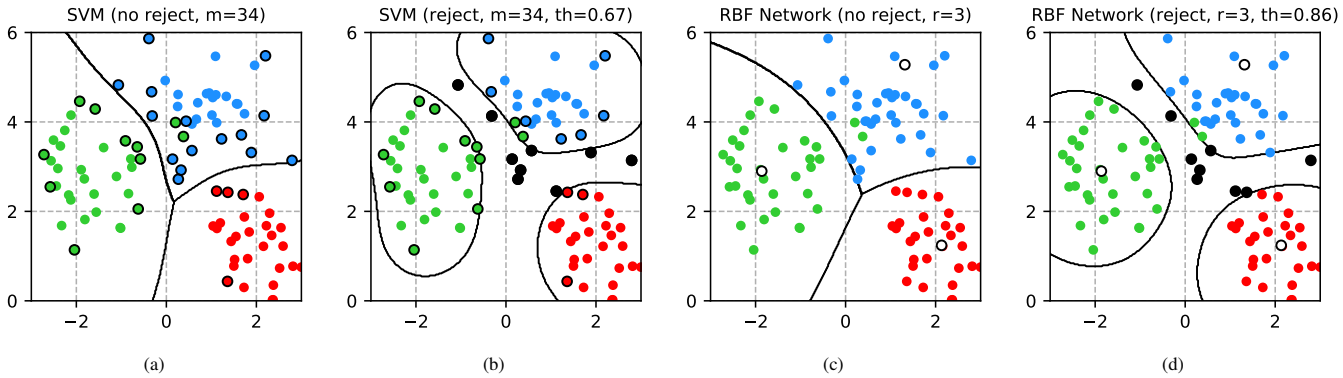

Figure 2: Comparison of classifiers decision regions on a two-dimensional classification example with three classes (green, blue, and red points), using multiclass SVMs with RBF kernels (SVM) and RBF Networks. a) SVM without reject option, the solution found exploits $m = 34$ support vectors (circled in black). b) SVM with reject option using a threshold $th = 0.67$ to obtain 10% FPR, rejected samples are highlighted with black dots. c) RBF Network without rejecting option, the solution found properly separates all classes using only $r = 3$ bases (black circles). d) RBF Network with reject option using a threshold $th = 0.86$ to obtain 10% FPR, rejected samples are highlighted with black dots. Notably, $r = 3$ is the minimum number of bases to ensure each class is correctly enclosed.

rejected if the prediction values of the combiner do not exceed a predefined threshold, which is tuned on a validation set not to exceed a given fraction of false rejections (i.e., natural samples detected as adversarial).

4. Adversarial Robustness Evaluation

A correct evaluation of proposed detection methods against adversarial examples is essential [9, 30], and it is not sufficient to evaluate such defenses against previous defense-unaware attacks that are likely to fail (see, e.g. [41, 42, 18]), leading to overly optimistic results in term of classifier robustness. To perform a fair defense evaluation, attacks should take into account the defense mechanism. Under this condition, many defenses were shown not to be as effective as claimed [29, 43, 30]. For instance, many defenses take advantage of *gradient masking*, i.e. they learn functions which are harder to optimize for gradient-based attacks; however, they can be easily bypassed by constructing smoother, differentiable approximations of their functions, e.g., learning surrogate models [9, 11, 44, 45] or replacing network layers which obfuscate gradients with smoother mappings [29, 43, 30].

When a defense exploits rejection, a defense-unaware attack may craft adversarial examples belonging to rejection regions, making it very difficult to evade such defense (Fig. 3). To perform a fair robustness evaluation of the proposed defense method an adaptive defense-aware attack is required.

In this work we adopt the security evaluation procedure proposed in [9, 30, 46], constructing *security evaluation curves* which show how the accuracy of attacked systems degrades under attacks crafted with increasing strength, i.e. the amount of perturbation. The more robust a defense method is, the more gracefully the curve decreases. It is remarkably important that accuracy curves should reach zero under a sufficiently large perturbation, for all evaluated defense - in an extreme ideal case, with an unbounded perturbation, the attacker can replace the source sample with a sample of another class. If this does not happen, then it may be that the attack algorithm is not able to perform correctly the optimization, and this in turn means that we are probably overestimating the defense robustness.

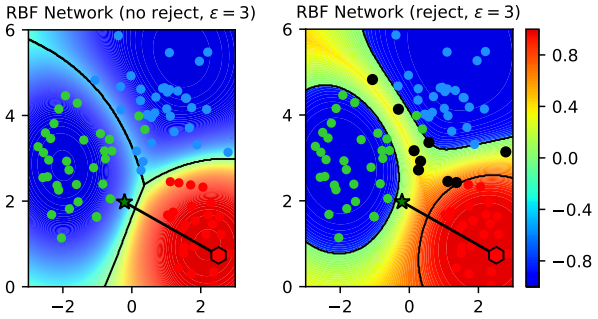

Figure 3: Conceptual representation on a 3-class bi-dimensional classification problem of the evasion attack against an RBF Network without (left) and with reject-based defense (right). The initial sample x_0 (red hexagon) has been modified to obtain the adversarial sample $x^\star$ (green star) which is wrongly recognized as an observation from the green class by the standard classifier, while it is correctly rejected when employing our defense. The values of the attack objective $\Omega(\boldsymbol{x})$ are shown in the background for both cases.

We formulate here an adaptive white-box attack suitable for all defenses considered in this work, and that takes into account rejection. Starting from a sample $\boldsymbol{x}$, the attacker can compute a maximum-allowed ε-sized adversarial perturbation obtaining the adversarial example $\boldsymbol{x}^\star$, by solving the following constrained optimization problem:

$$\boldsymbol{x}^\star \in \arg\min_{\boldsymbol{x}':\|\boldsymbol{x}-\boldsymbol{x}'\|\leq\varepsilon} \Omega(\boldsymbol{x}) = s_y(\boldsymbol{x}') - \max_{j\notin\{0,y\}} s_j(\boldsymbol{x}') \qquad (3)$$

being $\boldsymbol{x}'$ the modified input sample, $\|\boldsymbol{x} - \boldsymbol{x}'\| \leq \varepsilon$ is an ℓ_p-norm constraint (typical norms used for crafting adversarial examples are ℓ_1, ℓ_2 and ℓ_∞, for which efficient projection algorithms exist [47]), $y \in Y$ is the true class, and 0 is the rejection class.

Intuitively, to perform an untargeted (error-generic) evasion the output of the true class must be minimized, and the output of one competing class (excluding the reject class) must be maximized. The resulting objective function is negative in case of successful evasion, and its absolute value increases with the increasing classification confidence on the competing class. The

Algorithm 1 PGD-LS: PGD-based Maximum-confidence Adversarial Examples with Exponential Line Search

Require: $\boldsymbol{x}_0$: the input sample; η_0: the initial step size; Π: a projection operator on the ℓ_p-norm constraint $\|\boldsymbol{x}_0 - \boldsymbol{x}'\| \leq \varepsilon$; $t > 0$: a small positive number to ensure convergence.
Ensure: $\boldsymbol{x}'$: the adversarial example.

$\boldsymbol{x}' \leftarrow \boldsymbol{x}_0$
repeat
  $\boldsymbol{x} \leftarrow \boldsymbol{x}'$
  $\boldsymbol{x}''_0 \leftarrow \Pi(\boldsymbol{x} - \eta_0 \nabla\Omega(\boldsymbol{x}))$
  for $k = 1$ **to** $k < 10$ **do**
    $\eta_k \leftarrow 2^k \eta_0$
    $\boldsymbol{x}''_k \leftarrow \Pi(\boldsymbol{x} - \eta_k \nabla\Omega(\boldsymbol{x}))$
    if $\Omega(\boldsymbol{x}''_k) < \Omega(\boldsymbol{x}''_{k-1})$ **then**
      $\eta' \leftarrow \eta_k$
    end if
  end for
  $\boldsymbol{x}' \leftarrow \Pi(\boldsymbol{x} - \eta' \nabla\Omega(\boldsymbol{x}))$
until $|\Omega(\boldsymbol{x}') - \Omega(\boldsymbol{x})| \leq t$
return $\boldsymbol{x}'$

algorithm does not simply search for a minimum-distance adversarial example, but it maximizes the confidence of the attack. Although in this work we focus only on untargeted attacks, the proposed formulation can be easily extended to account for targeted (error-specific) evasion, as in [15].

To solve the optimization problem above, we use a projected gradient descent (PGD) algorithm with a line search to optimize the step size, named PGD-LS (Algorithm 1). The initial step size η_0 is doubled for ten times, computing the objective function at each step and choosing the step size which minimizes the function. The selected step size is then used to update the point $\boldsymbol{x}'$. Choosing a different η at each step gives two main advantages:

- Speeding-up the optimization: using larger step sizes (when possible) allows us to reach the convergence with a reduced number of iteration steps.
- Escaping local minima which may hinder the optimization process, using the larger step sizes.

In fact, while attacking DNR, we found that the optimization often got stuck in local minima inside reject decision regions, where the objective function gradient reaches very small values close to zero. The magnitude of these gradients strongly depends on the value of the γ parameter of SVM-RBF classifiers used by DNR, which is a negative exponent used in the kernel computation that controls the shape of decision regions around training samples. Larger values of γ produce more complex decision regions and smaller gradient magnitude, smaller values of γ conversely produce smoother decision regions. Our proposed attack exploiting an adaptive step size turns out to be more effective than standard fixed-size step attacks, as shown in the experiments of Section 5.

5. Experimental Analysis

In this section, we empirically evaluate the security of the proposed FADER defense mechanism (i) against defense-aware adversarial examples, and (ii) in a black-box setting where the attacker is essentially unaware of the defense mechanism used to protect the DNN classifier. After detailing our experimental setup (Sec. 5.1), we report the classifier's performance under attack by comparing it with NR and DNR detectors (Sec. 5.2). The aforementioned experiments are carried out on MNIST and CIFAR10 datasets. We also consider an additional set of experiments to validate the effectiveness of the proposed adaptive attack (PGD-LS) against standard PGD attacks in Sec. 5.3. In this case, we also consider ImageNet data, to show that our defensive approach can also be reliably used on larger state-of-the-art DNN classifiers. All these experiments are run using `secml` [48], i.e., a Python framework that enables benchmarking of attacks and defenses for secure machine learning. We plan to extend `secml` soon to include an implementation of the FADER-based detectors presented in this work.

5.1. Experimental Setup

We discuss here the datasets we use to evaluate our defense method and the classifiers we compare the performance with. All experiments are performed on a workstation equipped with an Intel(R) Xeon(R) CPU E5-2670 v3 @ 2.30GHz and an NVIDIA Quadro M6000 graphics card. The undefended DNNs run on the GPU, while all the attached detectors are executed on the CPU.

Datasets. Our analysis is performed on three image recognition datasets: MNIST, CIFAR10 and a ten class subset of ImageNet [49], referred to as *ImageNet10*. MNIST consists of 60,000 training and 10,000 test gray-scale samples of shape 28x28. CIFAR10 consists of 50,000 training and 10,000 test RBG samples of shape 32x32. ImageNet10 dataset is obtained by extending the Imagenette[3] dataset with a disjoint set of samples from the ImageNet validation split, so to compensate the lack of a publicly available test split for Imagenette. In particular, we selected 50 images per class coming from the ImageNet validation split, for a total of 500 test samples. ImageNet RBG samples are resized to 256x256 pixels and center-cropped to 224x224 pixels. For all datasets, images are normalized in $[0, 1]$ by dividing the input pixel values by 255.

Classifiers. We compare the performance of an undefended DNN (i.e. not implementing any rejection mechanism), which represents our baseline, with NR and DNR defense methods and their *fast* variants, i.e. FADER technique is applied. To implement the undefended DNNs for the MNIST dataset, we use the same architecture suggested by Carlini et al. [50]. For CIFAR10, instead, we consider a lightweight network that, despite its size, allows obtaining high performances. The two architectures under consideration are shown in Tables 2 and 3. For ImageNet10 we rely on the pre-trained AlexNet [51] DNN,

[3] `https://github.com/fastai/imagenette`

Id	Layer Type	Dimension
`relu1`	Conv. + ReLU	64 filters (5x5)
`relu2`	**Conv. + ReLU**	**64 filters (3x3)**
`relu3`	**Conv. + ReLU**	**64 filters (3x3)**
`relu4`	**Fully Connected + ReLU**	**32 units**
`dropout`	Dropout ($p = 0.5$)	
`softmax`	Softmax	10 units

Table 2: Model architecture of the MNIST Neural Network [50]. The layers used by DNR and FADER detectors are highlighted in bold.

Id	Layer Type	Dimension
`relu1`	Conv. + Batch Norm. + ReLU	64 filters (3x3)
`relu2`	Conv. + Batch Norm. + ReLU	64 filters (3x3)
`drop1`	Max Pooling + Dropout ($p = 0.1$)	2x2
`relu3`	Conv. + Batch Norm. + ReLU	128 filters (3x3)
`relu4`	Conv. + Batch Norm. + ReLU	128 filters (3x3)
`drop2`	Max Pooling + Dropout ($p = 0.2$)	2x2
`relu5`	Conv. + Batch Norm. + ReLU	256 filters (3x3)
`relu6`	Conv. + Batch Norm. + ReLU	256 filters (3x3)
`drop3`	**Max Pooling + Dropout (p = 0.3)**	2x2
`relu7`	**Conv. + Batch Norm. + ReLU**	512 filters (3x3)
`drop4`	Max Pooling + Dropout ($p = 0.4$)	2x2
`linear`	**Fully Connected**	512 units
`softmax`	Softmax	10 units

Table 3: Model architecture of the CIFAR10 Neural Network. The layers used by DNR and FADER detectors are highlighted in bold.

available on TorchVision.[4] We keep only outputs related to the ten classes of interest. As for the detectors, we consider the single-layer rejection mechanism on top of the pre-softmax activation layer, in the form of an SVM-RBF for NR [15] and the DNR defense approach [16] employing SVMs with RBF kernel as both layer detectors and the top combiner. For both NR and DNR we provide *fast* variants, employing size-controlled RBF networks, denoted as NR-RBF and DNR-RBF, respectively. As in previous work [16], we limit the number of layers inspected by the deep detectors to three. In particular, we consider the last three layers for the MNIST network; the last layer, the last batch-norm layer, and the second-to-last max-pooling layer for the CIFAR10 network (choice aimed at obtaining a reasonable amount of features); and sixth last, fourth last and third last layers for the AlexNet network on ImageNet10. We highlight the selected layers for MNIST and CIFAR10 networks in bold in Tables 2 and 3.

Training-test Splits. For MNIST and CIFAR10, we assume the DNNs to be pre-trained on separate training sets of 30,000 and 40,000 samples, respectively. The rest of the data of MNIST and CIFAR10 datasets is then used for training the NR and DNR detectors in both the SVM-based and RBF neurons-based configurations. As a pre-trained network, the ImageNet10 DNN is trained on the whole ImageNet training set. As such, we exploit the Imagenette samples to train NR and DNR detectors, and the overmentioned additional test split as test data. To avoid intersections between train and test data, we exclude from the Imagenette dataset a small amount of images that were also among ImageNet validation split.

For all datasets, we average the security evaluation results on five different runs. In each run, we consider 10,000 training samples and 1000 test sample (500 for ImageNet10), randomly drawn from the corresponding datasets. As previously done in [16], to avoid overfitting, the DNR combiner is trained by concatenating the outputs of the base SVMs detectors computed on separate validation sets, extracted from the training set using a 3-fold cross-validation procedure. This procedure is known as *stacked generalization* [25].

Parameter	MNIST	CIFAR10
Learning Rate	0.1	0.01
Momentum	0.9	0.9
Dropout	0.5	(see Table 3)
Batch Size	128	100
Epochs	50	75

Table 4: Parameters used to train the MNIST and CIFAR10 DNNs.

Parameter Setting. Table 4 reports the hyperparameters used for MNIST and CIFAR10 DNN's training. For all the three datasets, DNR detectors' best configuration is looked for in $C \in \{10^{-2}, \ldots, 10^{2}\}$ and $\gamma \in \{10^{-4}, \ldots, 10^{2}\}$ by performing a 3-fold cross-validation procedure to maximize accuracy on unperturbed training data. DNR optimal configurations for each dataset are reported in Table 5.

As DNR layer classifiers and combiners are not independently optimized during training, the NR best configuration can be obtained by lookup Table 5 for the layer of interest.

The architectures of FADER-based solutions are designed to maximize prototype reduction rate, while achieving comparable performances on clean test samples (see Tables 6 and 7). In terms of training, RBF-based neurons are optimized using `pytorch`[5] Adam optimizer with default settings for 250 epochs. Rejection threshold θ is set, in all the considered cases, to reject 10% of the samples when no attack is performed (at $\varepsilon = 0$).

Security Evaluation. We compare the aforementioned undefended neural networks and the rejection-based architectures in terms of their security evaluation curves [9], reporting classification accuracy against an increasing ℓ_2-norm perturbation size, used to perturb all the test samples, for MNIST and CIFAR10 datasets. In particular, under attack (i.e., for $\varepsilon > 0$), all the tested samples are adversarial examples and are considered correctly classified if either assigned to the rejection class or to their true class. We consider a significant interval of $\varepsilon \in [0, 5]$ and $\varepsilon \in [0, 2]$ for the attacks against MNIST and CIFAR10 datasets, respectively. The rejection rates, computed by dividing the number of rejected samples by the number of test samples, are also reported for all defense-aware classifiers (in absence of defense, the rejection rate will be zero). It is worth noting that, under this setting, the fraction of adversarial

[4] `https://pytorch.org/vision`

[5] `https://pytorch.org`

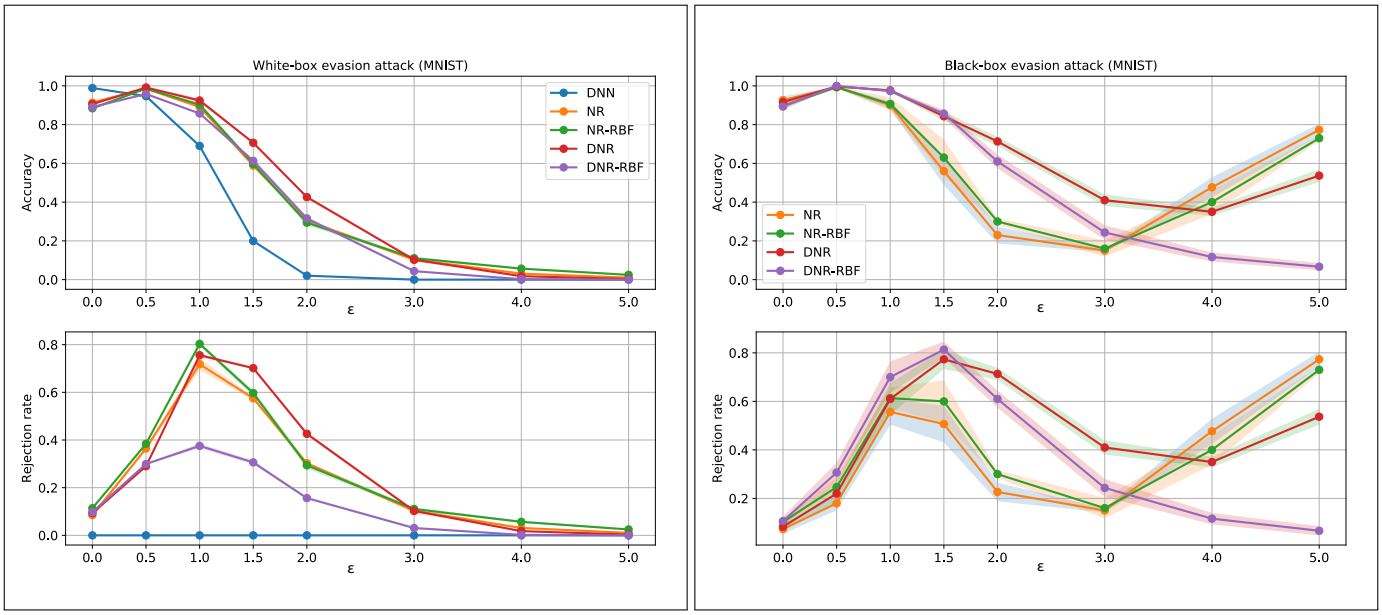

Figure 4: Security evaluation curves for MNIST data, under white-box (left) and black-box (right) settings. Mean accuracy at increasing ℓ_2-norm perturbation size is reported in the top subplots, while the bottom subplots show the corresponding rejection rates.

MNIST		
Layer	C	γ
`relu2`	10	1e-3
`relu3`	10	1e-2
`relu4`	1.0	1e-2
`combiner`	1e-1	1.0

CIFAR10		
Layer	C	γ
`drop3`	10	1e-3
`relu7`	1.0	1e-3
`linear`	1.0	1e-2
`combiner`	1e-4	1.0

ImageNet10		
Layer	C	γ
`clf1`	10	1e-4
`clf3`	1	1e-4
`clf4`	1	1e-4
`combiner`	1	1e-1

Table 5: DNR configurations for MNIST (top-left), CIFAR10 (top-right) and ImageNet10 (bottom) datasets.

examples at each $\varepsilon > 0$ which are correctly assigned to their true class is given by the difference between the accuracy and the rejection rate. For ImageNet10, we perform an evaluation at fixed $\varepsilon = 1.0$, in a white-box setting as a representative value for evaluating detectors' performance under attack.

5.2. Experimental Results

Adversarial Robustness Evaluation. The security evaluation curves for both white-box (defense-aware) and black-box settings, against MNIST and CIFAR10 datasets, are reported in Figures 4 and 5, top subplots. In the bottom subplots, we report the corresponding rejection rates at increasing ℓ_2-norm perturbation size. We also report in Table 10 the performance of all classifiers for ImageNet10 dataset against our attack (PGD-LS), at distance $\varepsilon = 1.0$. In the absence of an attack ($\varepsilon = 0$), the undefended DNN slightly outperforms all rejection-based detectors, due to a portion of samples that are incorrectly flagged as positive. Under attack ($\varepsilon > 0$), all detectors show improved robustness to adversarial examples compared to the standard DNN, as their accuracy decreases more gracefully. Notably, the performance of all detectors even increases for low values of ε, as the slightly modified testing images immediately become blind-spot adversarial examples, ending up in a region that is far from the rest of the data. As the input perturbation increases, such samples are gradually drifted inside a different class, making them indistinguishable for rejection-based defenses [15]. Interestingly, for the MNIST case, we notice a similar increase of accuracy for NR, DNR, and NR-RBF detectors tested in black-box setting at the highest values of ε. Due to the imperfect attacker's knowledge of the target classifier, when the ℓ_2 distance is large, the adversarial samples end up again once farther from the rest of the data and are detected by the defenses. This behavior is also confirmed by the corresponding rejection rate, which increases jointly with ε. By comparing the average adversarial robustness of the considered detectors, we show that FADER provide comparable effectiveness against adversarial examples as original solutions.

Prototype Reduction. Tables 6, 7, and 8 report for MNIST, CIFAR10 and ImageNet10 datasets, respectively, the number of prototypes employed in each detector component along with the estimated prototype reduction rate for FADER-based defenses. Notably, up to a 73× prototype reduction rate can be achieved for NR and 20× for DNR without performance drops on both natural and adversarial data for MNIST dataset. The same holds for the other considered datasets: FADER detector variants achieve up to a 50× and 82× prototype reduction rate against original detectors for CIFAR10 and ImageNet10, respectively. This prototype reduction rate has a substantial

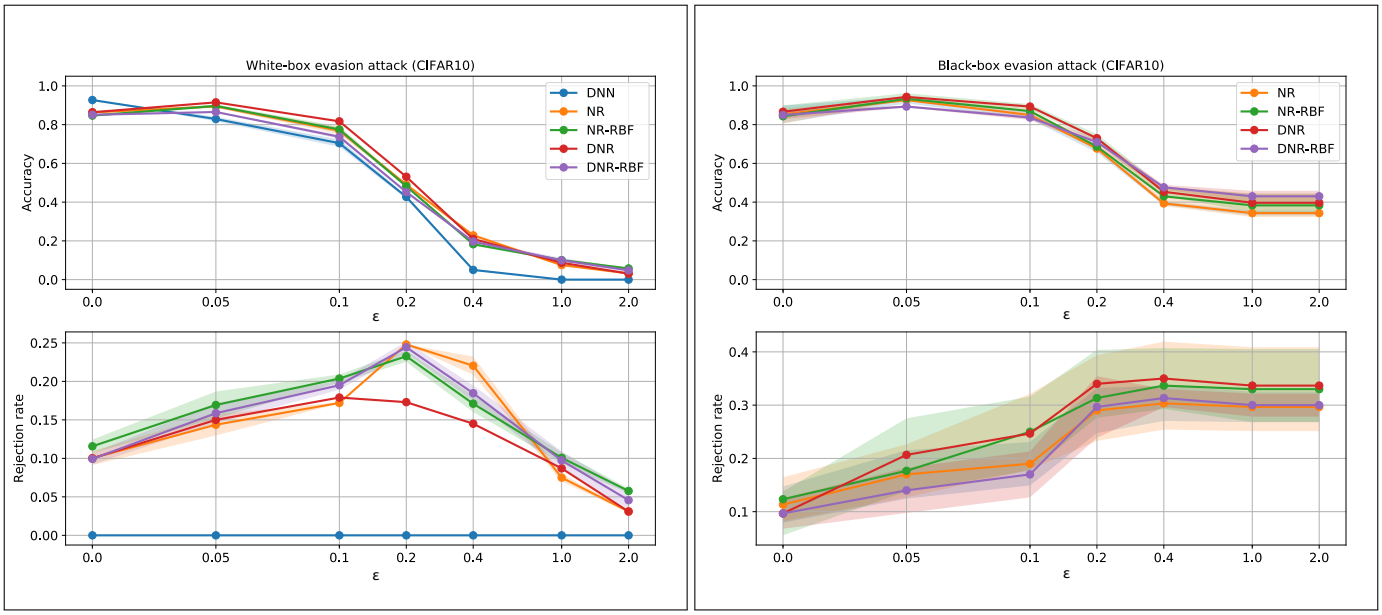

Figure 5: Security evaluation curves for CIFAR10 data, under white-box (left) and black-box (right) settings. Mean accuracy at increasing ℓ_2-norm perturbation size is reported in the top subplots, while the bottom subplots show the corresponding rejection rates.

impact in terms of space complexity; in particular, if we only consider the reference prototypes that are stored in memory to perform classification, FADER variants allow us to save a considerable amount of space. We quantify this in Tables 6, 7, and 8, considering a (conservative) memory consumption estimate of just 8 bytes for each floating-point number.

Runtime Complexity. To effectively quantify the speedup induced at runtime by FADER variants over the SVM-based detectors, a benchmark evaluation is considered here. Figure 6 reports the time spent by the undefended DNN and each detector, executed on our workstation (as detailed in the experimental setup), to classify an increasing number of test samples, averaged over 10 repetitions, for each dataset. As detailed in Section 3, the runtime complexity scales linearly with the test set size. The results for the last point of the curve, corresponding to 5000 test samples, are also reported in Table 9 (standard deviation in brackets), to better quantify the time reduction. The expected speedup due to a reduced number of prototypes is confirmed by the time measurements on all the three datasets, reaching up to 3.46× on CIFAR10.

To conclude, even though the reduction of time complexity is not as large as the prototype reduction rates, FADER enables a significant reduction in space complexity, which makes it well suited to low-power, embedded devices or edge-cloud systems.

5.3. Comparison with PGD

In this section, we compare the performance of our attack PGD-LS (Algorithm 1) against the standard Projected Gradient Descent (PGD) with normalized step proposed in [52]. The performance under white-box (defense-aware) attack, against MNIST, CIFAR10 and ImageNet10 datasets, using 1000 test samples (500 for ImageNet10), is reported in Table 10. It is worth pointing out that PGD-LS outperforms PGD especially when optimizing attacks becomes more challenging, e.g., due to the presence of wide flat local optima which are difficult to escape if the step size is not sufficiently large. This happens when attacking NR and NR-BRF on MNIST, and DNR on MNIST and CIFAR10. In these cases, PGD would only give a false sense of security (i.e., induce one to think that such methods are more robust), whereas PGD-LS successfully evades them with a higher probability, providing a more reliable robustness evaluation [30, 53].

As an illustration, we report in Figure 7 few ImageNet samples obtained using the two considered attack algorithms against the different detectors, together with the relative induced adver-

Detector	# prototypes							Accuracy
	`relu2`	`relu3`	`relu4`	`combiner`	`total`	`reduction`	memory	
NR	-	-	736	-	736	-	188 KB	0.984
NR-RBF	-	-	**10**	-	**10**	**~ 73×**	**3 KB**	**0.985**
DNR	2304	2375	736	9152	11527	-	43 MB	0.961
DNR-RBF	**250**	**250**	**50**	**10**	**560**	**~ 20×**	**4** MB	**0.989**
# features	1600	576	32	30				

Table 6: Comparison of the number of prototypes used by each component of the rejection-based defense architectures (FADER in bold) on MNIST dataset. We also report the mean accuracy of each detector at $\varepsilon = 0$ and the memory consumption related to the stored reference prototypes.

Detector	# prototypes							Accuracy
	`drop3`	`relu7`	`linear`	`combiner`	`total`	`reduction`	memory	
NR	-	-	5257	-	5275	-	22 MB	0.915
NR-RBF	-	-	**100**	-	**100**	**~ 50×**	**410 KB**	**0.911**
DNR	7198	3100	5257	10000	25555	-	311 MB	0.913
DNR-RBF	**500**	**300**	**100**	**100**	**1000**	**~ 28×**	**22 MB**	**0.892**
# features	4096	2048	512	30				

Table 7: Comparison of the number of prototypes used by each component of the rejection-based defense architectures (FADER in bold) on CIFAR10 dataset. We also report the mean accuracy of each detector at $\varepsilon = 0$ and the memory consumption related to the stored reference prototypes.

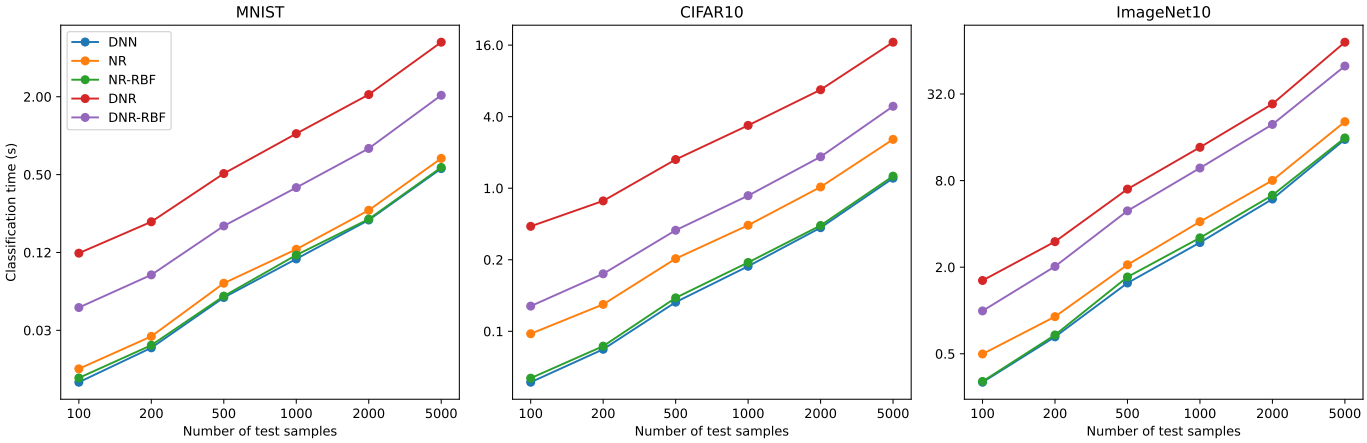

Figure 6: Classification time (in seconds) against an increasing number of test samples, averaged on 10 runs, using batches of 256 samples. Similar linear dependencies are also found for batch sizes of 32, 64, 128, 512 and 1024. FADER gives a consistent advantage over NR and DNR, as also quantified in Table 9.

Detector	# prototypes							Accuracy
	clf1	clf3	clf4	combiner	total	reduction	memory	
NR	-	-	4103	-	4103	-	134 MB	0.894
NR-RBF	-	-	**50**	-	**50**	**~ 82×**	**2 MB**	**0.882**
DNR	6729	6590	4103	1535	14854	-	571 MB	0.894
DNR-RBF	**200**	**200**	**100**	**50**	**550**	**~ 27×**	**6 MB**	**0.874**
# features	4096	4096	4096	30				

Table 8: Comparison of the number of prototypes used by each component of the rejection-based defense architectures (FADER in bold) on ImageNet10 dataset. We also report the mean accuracy of each detector at $\varepsilon = 0$ and the memory consumption related to the stored reference prototypes.

Classifier	MNIST	CIFAR10	ImageNet10
DNN	557 ± 4	1214 ± 1	15453 ± 280
NR	667 ± 6	2575 ± 5	20543 ± 750
NR-RBF	**568 ± 7 (1.17×)**	**1263 ± 1 (2.03×)**	**15843 ± 639 (1.30×)**
DNR	5281 ± 20	16927 ± 67	73336 ± 3324
DNR-RBF	**2053 ± 32 (2.57×)**	**4891 ± 121 (3.46×)**	**50069 ± 615 (1.46×)**

Table 9: Expected time ± standard deviation (in milliseconds) to classify 5000 samples, averaged over 10 runs. The reduction attained by each FADER detector (in bold), computed as the ratio between the time spent by the SVM-based detector and the corresponding FADER variant, is reported in parenthesis.

sarial perturbation: PGD-LS tends to produce finer perturbation than standard PGD attack, yet being effective (Table 10).

6. Related Work

The problem of countering adversarial attacks is far from being new. The first adversary-aware classification algorithm against evasion attacks has been proposed in 2004, which is based on simulating attacks and iteratively retraining the classifier on them [55]. More recently, similar techniques took the name of *adversarial training* and were employed to counter adversarial examples in DNNs [12, 13], or to harden decision trees and random forests [56]. Other variants of this approach [57, 58], instead of encouraging the correct labeling of the attacks by augmenting the training set, introduce a new class in the model solely for the adversarial attacks, and train the model to detect them. Bhagoji et al. [59] propose a similar defense

Classifier	MNIST ($\varepsilon = 1.5$)		CIFAR10 ($\varepsilon = 0.2$)		ImageNet10 ($\varepsilon = 1.0$)	
	PGD-LS	PGD	PGD-LS	PGD	PGD-LS	PGD
DNN	0.20	0.25	0.42	0.38	0.23	0.24
NR	0.60	0.82	0.46	0.43	0.41	0.43
NR-RBF	**0.59**	**0.84**	**0.46**	**0.40**	**0.40**	**0.51**
DNR	0.71	1.00	0.53	0.95	0.55	0.58
DNR-RBF	**0.62**	**0.74**	**0.45**	**0.40**	**0.55**	**0.56**

Table 10: Classification accuracy under white-box attack at fixed values of ε for the different classifiers (FADER in bold), MNIST data ($\varepsilon = 1.5$), CIFAR10 data ($\varepsilon = 0.2$) and ImageNet10 data ($\varepsilon = 1.0$), using our PGD with Line Search (PGD-LS, Algorithm 1), and a standard PGD with normalized step [54]. PGD-LS outperforms PGD when optimizing attacks is more challenging, e.g., when attacking NR and NR-BRF on MNIST, and DNR on MNIST and CIFAR10.

based on dimensionality reduction instead, which retrains the classifier on a D-dimensional version of the inputs, with D $\ll d$. This defense restricts the attacker to only manipulate the first D components, resulting in a dramatic magnitude increase of the perturbation required to produce an effective attack.

As many retraining-based techniques are founded on heuristics, with no formal guarantees on convergence and robustness properties, more structured approaches rely on game theory. Zero-sum games learn invariant transformations like feature insertion, deletion and rescaling [60, 61, 62]. *Robust optimization* is also formulated as a min-max (zero-sum) game, in which the inner problem maximizes the training loss by manipulating the training points under worst-case, bounded perturbations, while the outer problem minimizes the corresponding worst-case training loss [13, 54]. A direct result derived from these techniques is the equivalence between regularized learning problems and robust optimization, under some linearity assumptions [63], which has enabled approximating computationally demanding secure learning models, like the aforementioned ones based on game theory, with more efficient strategies based on regularizing the objective function in a specific manner [64]. The main effect of these methods is to smooth out the decision function of the classifier reducing the norm of the input gradients, making it less sensitive to worst-case input changes.

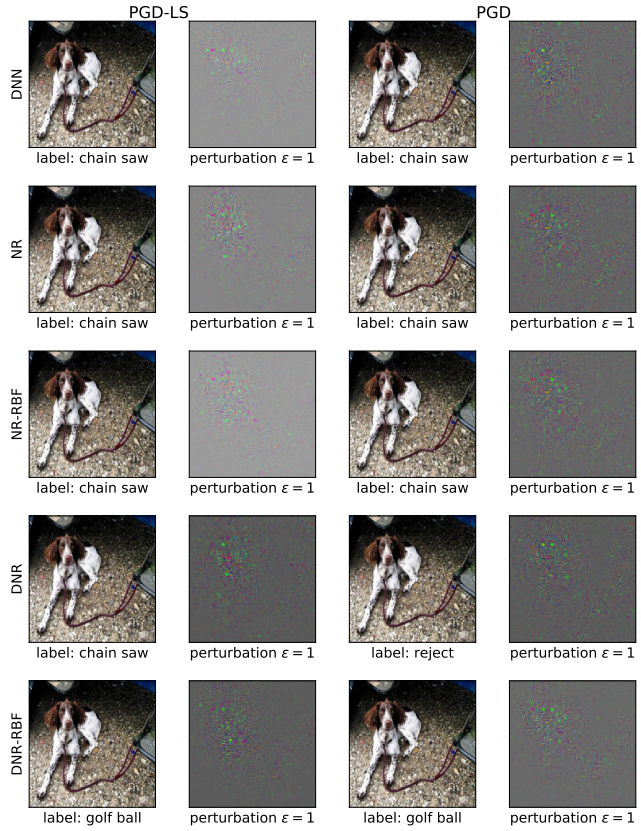

Figure 7: Perturbed samples from ImageNet10 dataset produced by attacking each classifier using PGD-LS (left columns) and PGD (right columns) algorithms. The maximum size of the ℓ_2 perturbation is equally set to $\varepsilon = 1$.

To achieve this, few works also proposed to improve the so-called *evenness* of the classifier's parameters against sparse attacks [65, 66, 67]. Other work considered more complicated game-theoretical approaches based on non-zero sum games, in which the attacker and the defender may have different objective functions. This work introduced Nash and Stackelberg games for secure learning, under formal conditions for existence and uniqueness of the game equilibrium, under the assumption that each player completely knows the game details and the other players' payoff functions [68, 69]. Randomized games were also considered in [70], to account for imperfect knowledge and uncertainty on the players' strategies [71]. Regrettably, however, machine learning in adversarial scenarios is not a board game with well-defined rules, thus understanding the extent to which the resulting attack strategies are representative of practical scenarios remains an open issue [72, 73]. Also, the scalability of these methods and, generally, of adversarial retraining procedures, to large datasets and high-dimensional feature spaces is in doubt, as it may be too computationally costly to generate a sufficient number of attack samples to correctly represent their distribution.

Other defenses try to detect adversarial examples by comparing the distribution of legitimate samples to the distribution of attacks. In [57] a classical statistical method is exploited, Maximum Mean Discrepancy (MMD), which allows to determine if two set of samples are drawn from the same underlying distribution. Hendrycks & Gimpel [74] propose the use of Principal Component Analysis (PCA) by arguing that adversarial samples place a higher (lower) weight on the later (earlier) principal components with respect to legitimate samples. Li et al. [75], instead, apply PCA to the values after inner convolutional layers of a CNN model, and use a *cascade classifier* to detect the differences between the two distributions. Similarly, in [76] a Gaussian Mixture Model is used to analyze if the outputs of the model in case of adversarial examples belong to a distribution different than that of legitimate samples.

Finally, recent efforts combine the advantages of adversarial training and detection mechanisms [77], reporting promising results. However, the problem remains still challenging and largely unsolved, especially when it comes to evaluating adversarial robustness against adaptive white-box attacks [78].

7. Conclusions and Future Work

In this work, we presented FADER (*Fast Adversarial Example Rejection*), i.e., a technique to speed up rejection-based defenses against adversarial examples. FADER exploits RBF networks to control the number of reference prototypes required for predictions, resulting in accuracy vs. detection time efficiency gain. In our experiments, we demonstrated a 73× prototypes reduction with respect to analyzed detectors for MNIST dataset, up to 50× prototypes reduction for CIFAR10 image recognition task, and up to 82× in case of the ImageNet10 dataset, while maintaining comparable performance on both clean and adversarial data. This can have a strong impact on real-world scenarios involving adversarial-example detection on low-capability (e.g., edge) devices. We further provided a comprehensive review of multiple detector-based adversarial detection techniques from the literature, framing them in the form of a proposed adversarial-example detection framework designed to accommodate both existing and newer methods to come (Section 2). To properly evaluate FADER's response to adversarial attacks, we designed a novel attack algorithm that takes into account the defense to not overestimate performances under attack (see Section 4). Experimental results on different image classification tasks highlight FADER-based defenses as more *efficient* solutions than original ones in terms of required prototypes while maintaining comparable performances both on clean data and under attack. As for future work, we aim to improve FADER performance under attack by employing proper input gradient regularization [64, 79] and to test novel FADER architectural variants to further reduce the computational overhead of adversarial-example detection schemes.

Acknowledgements

This work has been partially supported by the PRIN 2017 project RexLearn (grant no. 2017TWNMH2), funded by the Italian Ministry of Education, University and Research; and by BMK, BMDW, and the Province of Upper Austria in the frame of the COMET Programme managed by FFG in the COMET Module S3AI.

References

[1] K. He, X. Zhang, S. Ren, J. Sun, Deep Residual Learning for Image Recognition, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016. doi:10.1109/CVPR.2016.90. arXiv:1512.03385.

[2] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, in: Advances in Neural Information Processing Systems, 2017.

[3] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, D. Hassabis, Mastering the game of Go with deep neural networks and tree search, Nature (2016).

[4] M. Bojarski, D. D. Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, K. Zieba, End to end learning for self-driving cars, CoRR abs/1604.07316 (2016).

[5] D. Gibert, C. Mateu, J. Planes, The rise of machine learning for detection and classification of malware: Research developments, trends and challenges, Journal of Network and Computer Applications 153 (2020) 102526.

[6] L. R. Carlos-Roca, I. H. Torres, C. F. Tena, Facial recognition application for border control, in: 2018 International Joint Conference on Neural Networks (IJCNN), 2018, pp. 1–7. doi:10.1109/IJCNN.2018.8489113.

[7] A. R. Vaka, B. Soni, S. R. K., Breast cancer detection by leveraging machine learning, ICT Express (2020).

[8] L. Huang, A. D. Joseph, B. Nelson, B. Rubinstein, J. D. Tygar, Adversarial machine learning, in: 4th ACM Workshop on Artificial Intelligence and Security (AISec 2011), Chicago, IL, USA, 2011, pp. 43–57.

[9] B. Biggio, F. Roli, Wild patterns: Ten years after the rise of adversarial machine learning, Pattern Recognition 84 (2018) 317–331.

[10] B. Biggio, B. Nelson, P. Laskov, Poisoning attacks against support vector machines, in: J. Langford, J. Pineau (Eds.), 29th Int'l Conf. on Machine Learning, Omnipress, 2012, pp. 1807–1814.

[11] B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Šrndić, P. Laskov, G. Giacinto, F. Roli, Evasion attacks against machine learning at test time, in: H. Blockeel, K. Kersting, S. Nijssen, F. Železný (Eds.), Machine Learning and Knowledge Discovery in Databases (ECML PKDD), Part III, volume 8190 of *LNCS*, Springer Berlin Heidelberg, 2013, pp. 387–402.

[12] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, R. Fergus, Intriguing properties of neural networks, in: International Conference on Learning Representations, 2014. URL: http://arxiv.org/abs/1312.6199.

[13] I. J. Goodfellow, J. Shlens, C. Szegedy, Explaining and harnessing adversarial examples, in: International Conference on Learning Representations, 2015.

[14] R. Feinman, R. R. Curtin, S. Shintre, A. B. Gardner, Detecting adversarial samples from artifacts, arXiv preprint arXiv:1703.00410 (2017).

[15] M. Melis, A. Demontis, B. Biggio, G. Brown, G. Fumera, F. Roli, Is deep learning safe for robot vision? Adversarial examples against the iCub humanoid, in: ICCVW Vision in Practice on Autonomous Robots (ViPAR), IEEE, 2017, pp. 751–759.

[16] A. Sotgiu, A. Demontis, M. Melis, B. Biggio, G. Fumera, X. Feng, F. Roli, Deep neural rejection against adversarial examples, EURASIP J. Information Security 2020 (2020).

[17] N. Papernot, P. McDaniel, Deep k-Nearest Neighbors: Towards Confident, Interpretable and Robust Deep Learning (2018).

[18] D. Meng, H. Chen, MagNet: a two-pronged defense against adversarial examples, in: 24th ACM Conf. Computer and Comm. Sec. (CCS), 2017.

[19] F. Crecchi, D. Bacciu, B. Biggio, Detecting adversarial examples through nonlinear dimensionality reduction, in: ESANN '19, 2019.

[20] A. Lamb, J. Binas, A. Goyal, D. Serdyuk, S. Subramanian, I. Mitliagkas, Y. Bengio, Fortified Networks: Improving the Robustness of Deep Networks by Modeling the Manifold of Hidden Representations (2018).

[21] P. Samangouei, M. Kabkab, R. Chellappa, Defense-gan: Protecting classifiers against adversarial attacks using generative models, arXiv preprint arXiv:1805.06605 (2018).

[22] N. Papernot, P. D. McDaniel, I. J. Goodfellow, Transferability in machine learning: from phenomena to black-box attacks using adversarial samples, ArXiv e-prints abs/1605.07277 (2016).

[23] J. H. Metzen, T. Genewein, V. Fischer, B. Bischoff, On detecting adversarial perturbations, in: 5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings, 2017. arXiv:1702.04267.

[24] L. Van Der Maaten, G. Hinton, Visualizing Data using t-SNE, Technical Report, 2008. URL: http://www.jmlr.org/papers/volume9/vandermaaten08a/vandermaaten08a.pdf.

[25] D. H. Wolpert, Stacked generalization, Neural Networks 5 (1992) 241–259.

[26] C. Saunders, A. Gammerman, V. Vovk, Transduction with confidence and credibility, in: IJCAI International Joint Conference on Artificial Intelligence, 1999.

[27] V. Vovk, A. Gammerman, C. Saunders, Machine-learning applications of algorithmic randomness, in: Proceedings of the Sixteenth International Conference on Machine Learning, ICML '99, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1999, p. 444–453.

[28] G. Shafer, V. Vovk, A tutorial on conformal prediction, Journal of Machine Learning Research (2008).

[29] N. Carlini, D. A. Wagner, Adversarial examples are not easily detected: Bypassing ten detection methods, in: B. M. Thuraisingham, B. Biggio, D. M. Freeman, B. Miller, A. Sinha (Eds.), 10th ACM Workshop on Artificial Intelligence and Security, AISec '17, ACM, New York, NY, USA, 2017, pp. 3–14.

[30] A. Athalye, N. Carlini, D. A. Wagner, Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples, in: ICML, volume 80 of *JMLR Workshop and Conference Proceedings*, JMLR.org, 2018, pp. 274–283.

[31] L. De Alfaro, Neural Networks with Structural Resistance to Adversarial Attacks, Technical Report, 2018. URL: https://github.com/lucadealfaro/rbfi. arXiv:1809.09262v1.

[32] P. Habib Zadeh, R. Hosseini, S. Sra, Deep-RBF Networks Revisited: Robust Classification with Rejection, Technical Report, 2019. arXiv:1812.03190v1.

[33] J. Chenou, G. Hsieh, T. Fields, Radial basis function network: Its robustness and ability to mitigate adversarial examples, in: Proceedings - 6th Annual Conference on Computational Science and Computational Intelligence, CSCI 2019, Institute of Electrical and Electronics Engineers Inc., 2019, pp. 102–106. doi:10.1109/CSCI49370.2019.00024.

[34] A. Bendale, T. E. Boult, Towards open set deep networks, in: IEEE Conf. Computer Vision and Patt. Rec., 2016, pp. 1563–1572.

[35] W. J. Scheirer, A. Rocha, R. Michaels, T. E. Boult, Meta-recognition: The theory and practice of recognition score analysis, IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI) 33 (2011) 1689–1695.

[36] M. Melis, L. Piras, B. Biggio, G. Giacinto, G. Fumera, F. Roli, Fast image classification with reduced multiclass support vector machines, in: Image Analysis and Processing—ICIAP 2015, Springer, 2015, pp. 78–88.

[37] I. Steinwart, Sparseness of support vector machines, J. Mach. Learn. Res. 4 (2003) 1071–1105.

[38] O. Chapelle, Training a support vector machine in the primal, Neural Comput. 19 (2007) 1155–1178.

[39] A. Demontis, M. Melis, B. Biggio, G. Fumera, F. Roli, Super-sparse learning in similarity spaces, IEEE Computational Intelligence Magazine 11 (2016) 36–45.

[40] M. Claesen, F. D. Smet, J. A. K. Suykens, B. D. Moor, Fast prediction with svm models containing rbf kernels, 2014. arXiv:1403.0736.

[41] J. Lu, T. Issaranon, D. Forsyth, Safetynet: Detecting and rejecting adversarial examples robustly, in: The IEEE International Conference on Computer Vision (ICCV), 2017.

[42] N. Papernot, P. McDaniel, X. Wu, S. Jha, A. Swami, Distillation as a defense to adversarial perturbations against deep neural networks, in: 2016 IEEE Symposium on Security and Privacy (SP), 2016, pp. 582–597. doi:10.1109/SP.2016.41.

[43] N. Carlini, D. A. Wagner, Towards evaluating the robustness of neural networks, in: IEEE Symp. Security and Privacy, IEEE CS, 2017, pp. 39–57.

[44] P. Russu, A. Demontis, B. Biggio, G. Fumera, F. Roli, Secure kernel machines against evasion attacks, in: 9th ACM Workshop on Artificial Intelligence and Security, AISec '16, ACM, New York, NY, USA, 2016, pp. 59–69.

[45] M. Melis, D. Maiorca, B. Biggio, G. Giacinto, F. Roli, Explaining black-box android malware detection, in: 26th European Signal Processing

Conf., EUSIPCO, IEEE, Rome, Italy, 2018, pp. 524–528.
[46] B. Biggio, G. Fumera, F. Roli, Security evaluation of pattern classifiers under attack, IEEE Trans. Knowl. and Data Eng. 26 (2014) 984–996.
[47] J. Duchi, S. Shalev-Shwartz, Y. Singer, T. Chandra, Efficient projections onto the l1-ball for learning in high dimensions, in: Proc. 25th ICML, ICML '08, ACM, New York, NY, USA, 2008, pp. 272–279.
[48] M. Melis, A. Demontis, M. Pintor, A. Sotgiu, B. Biggio, secml: A Python Library for Secure and Explainable Machine Learning (2019).
[49] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, L. Fei-Fei, ImageNet: A Large-Scale Hierarchical Image Database, in: CVPR09, 2009.
[50] N. Carlini, D. Wagner, Adversarial examples are not easily detected: Bypassing ten detection methods, in: Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security, 2017, pp. 3–14.
[51] A. Krizhevsky, One weird trick for parallelizing convolutional neural networks, ArXiv abs/1404.5997 (2014).
[52] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, A. Vladu, Towards deep learning models resistant to adversarial attacks, in: ICLR, 2018.
[53] N. Carlini, A. Athalye, N. Papernot, W. Brendel, J. Rauber, D. Tsipras, I. Goodfellow, A. Madry, A. Kurakin, On evaluating adversarial robustness, ArXiv e-prints 1902.06705 (2019).
[54] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, A. Vladu, Towards deep learning models resistant to adversarial attacks, in: International Conference on Learning Representations, 2018.
[55] N. Dalvi, P. Domingos, S. Sanghai, D. Verma, Adversarial classification, in: 10th ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining, 2004, pp. 99–108.
[56] A. Kantchelian, J. D. Tygar, A. D. Joseph, Evasion and hardening of tree ensemble classifiers, in: 33rd ICML, volume 48 of *JMLR Workshop and Conference Proceedings*, JMLR.org, 2016, pp. 2387–2396.
[57] K. Grosse, P. Manoharan, N. Papernot, M. Backes, P. McDaniel, On the (statistical) detection of adversarial examples, arXiv preprint arXiv:1702.06280 (2017).
[58] Z. Gong, W. Wang, W.-S. Ku, Adversarial and clean data are not twins, arXiv preprint arXiv:1704.04960 (2017).
[59] A. N. Bhagoji, D. Cullina, P. Mittal, Dimensionality reduction as a defense against evasion attacks on machine learning classifiers, arXiv preprint arXiv:1704.02654 2 (2017).
[60] A. Globerson, S. T. Roweis, Nightmare at test time: robust learning by feature deletion, in: W. W. Cohen, A. Moore (Eds.), Proceedings of the 23rd International Conference on Machine Learning, volume 148, ACM, 2006, pp. 353–360.
[61] C. H. Teo, A. Globerson, S. Roweis, A. Smola, Convex learning with invariances, in: J. Platt, D. Koller, Y. Singer, S. Roweis (Eds.), Advances in Neural Information Processing Systems 20, MIT Press, Cambridge, MA, 2008, pp. 1489–1496.
[62] O. Dekel, O. Shamir, L. Xiao, Learning to classify with missing and corrupted features, Machine Learning 81 (2010) 149–178. 10.1007/s10994-009-5124-8.
[63] H. Xu, C. Caramanis, S. Mannor, Robustness and regularization of support vector machines, J. Mach. Learn. Res. 10 (2009) 1485–1510.
[64] A. Demontis, M. Melis, B. Biggio, D. Maiorca, D. Arp, K. Rieck, I. Corona, G. Giacinto, F. Roli, Yes, machine learning can be more secure! a case study on android malware detection, IEEE Trans. on Dependable and Secure Computing 16 (2019) 711–724.
[65] A. Kolcz, C. H. Teo, Feature weighting for improved classifier robustness, in: Sixth Conference on Email and Anti-Spam (CEAS), Mountain View, CA, USA, 2009.
[66] B. Biggio, G. Fumera, F. Roli, Multiple classifier systems for robust classifier design in adversarial environments, Int'l J. Mach. Learn. and Cybernetics 1 (2010) 27–41.
[67] M. Melis, M. Scalas, A. Demontis, D. Maiorca, B. Biggio, G. Giacinto, F. Roli, Do gradient-based explanations tell anything about adversarial robustness to android malware?, arXiv preprint arXiv:2005.01452 (2020).
[68] M. Brückner, C. Kanzow, T. Scheffer, Static prediction games for adversarial learning problems, J. Mach. Learn. Res. 13 (2012) 2617–2654.
[69] W. Liu, S. Chawla, Mining adversarial patterns via regularized loss minimization., Machine Learning 81 (2010) 69–83.
[70] S. Rota Bulò, B. Biggio, I. Pillai, M. Pelillo, F. Roli, Randomized prediction games for adversarial machine learning, IEEE Trans. on Neural Networks and Learning Systems 28 (2017) 2466–2478.
[71] M. Großhans, C. Sawade, M. Brückner, T. Scheffer, Bayesian games for adversarial regression problems, in: Journal of Machine Learning Research - Proc. 30th International Conference on Machine Learning (ICML), volume 28, 2013, pp. 55–63.
[72] M. Wooldridge, Does game theory work?, Intelligent Systems, IEEE 27 (2012) 76–80.
[73] G. Cybenko, C. E. Landwehr, Security analytics and measurements, IEEE Security & Privacy 10 (2012) 5–8.
[74] D. Hendrycks, K. Gimpel, Early methods for detecting adversarial images, arXiv preprint arXiv:1608.00530 (2016).
[75] X. Li, F. Li, Adversarial examples detection in deep networks with convolutional filter statistics, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 5764–5772.
[76] R. Feinman, R. R. Curtin, S. Shintre, A. B. Gardner, Detecting adversarial samples from artifacts, arXiv preprint arXiv:1703.00410 (2017).
[77] X. Yin, S. Kolouri, G. Rohde, Gat: Generative adversarial training for adversarial example detection and robust classification, in: ICLR, 2020.
[78] F. Tramer, N. Carlini, W. Brendel, A. Madry, On adaptive attacks to adversarial example defenses, in: H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, H. Lin (Eds.), Advances in Neural Information Processing Systems, volume 33, Curran Associates, Inc., 2020, pp. 1633–1645. URL: `https://proceedings.neurips.cc/paper/2020/file/11f38f8ecd71867b42433548d1078e38-Paper.pdf`.
[79] C.-J. Simon-Gabriel, Y. Ollivier, L. Bottou, B. Schölkopf, D. Lopez-Paz, First-order adversarial vulnerability of neural networks and input dimension, in: K. Chaudhuri, R. Salakhutdinov (Eds.), 36th ICML, volume 97 of *PMLR*, 2019, pp. 5809–5817.