Sotgiu *et al. EURASIP Journal on Information Security* (2020) 2020:5
https://doi.org/10.1186/s13635-020-00105-y

EURASIP Journal on Information Security

RESEARCH Open Access

Deep neural rejection against adversarial examples

Angelo Sotgiu[1], Ambra Demontis[1*], Marco Melis[1], Battista Biggio[1,2], Giorgio Fumera[1], Xiaoyi Feng[3] and Fabio Roli[1,2]

Abstract

Despite the impressive performances reported by deep neural networks in different application domains, they remain largely vulnerable to adversarial examples, i.e., input samples that are carefully perturbed to cause misclassification at test time. In this work, we propose a deep neural rejection mechanism to detect adversarial examples, based on the idea of rejecting samples that exhibit anomalous feature representations at different network layers. With respect to competing approaches, our method does not require generating adversarial examples at training time, and it is less computationally demanding. To properly evaluate our method, we define an adaptive white-box attack that is aware of the defense mechanism and aims to bypass it. Under this worst-case setting, we empirically show that our approach outperforms previously proposed methods that detect adversarial examples by only analyzing the feature representation provided by the output network layer.

Keywords: Adversarial machine learning, Deep neural networks, Adversarial examples

1 Introduction

Despite their impressive performances on a variety of tasks, it has been known for more than a decade that machine-learning algorithms can be misled by different adversarial attacks, staged either at training or at test time [1, 2]. After the first attacks proposed against linear classifiers in 2004 [3, 4], Biggio et al. [5, 6] have been the first to show that nonlinear machine learning algorithms, including support vector machines (SVMs) and neural networks, can be misled by gradient-based optimization attacks [1]. Nevertheless, such vulnerabilities of learning algorithms have become extremely popular only after that Szegedy et al. [7, 8] have demonstrated that also deep learning algorithms exhibiting superhuman performances on image classification tasks suffer from the same problems. They have shown that even only slightly manipulating the pixels of an input image can be sufficient to induce deep neural networks to misclassify its content. Such attacks have then been popularized under the name of *adversarial examples* [2, 6, 7].

Since the seminal work by Szegedy et al. [7, 8], many defense methods have been proposed to mitigate the threat of adversarial examples. Most of the proposed defenses have been shown to be ineffective against more sophisticated attacks (i.e., attacks that are aware of the defense mechanism), leaving the problem of defending neural networks against adversarial examples still open. According to [2], the most promising defenses can be broadly categorized into two families. The first includes approaches based on robust optimization and game-theoretical models [9–11]. These approaches, which also encompass *adversarial training* [8], explicitly model the interactions between the classifier and the attacker to learn robust classifiers. The underlying idea is to incorporate knowledge of potential attacks during training. The second family of defenses (complementary to the first) is based on the idea of rejecting samples that exhibit an outlying behavior with respect to unperturbed training data [12–16].

*Correspondence: ambra.demontis@unica.it
[1]DIEE, University of Cagliari, Piazza d'Armi, 09123 Cagliari, Italy
Full list of author information is available at the end of the article

In this work, we focus on defenses based on rejection mechanisms and try to improve their effectiveness. In fact, it has been shown that only relying upon the feature representation learned by the last network layer to reject adversarial examples is not sufficient [12, 13]. In particular, it happens that adversarial examples become indistinguishable from samples of the target class at such a higher representation level even for small input perturbations. To overcome this issue, we propose here a defense mechanism, named *deep neural rejection* (DNR), based on analyzing the representations of input samples at *different* network layers and on rejecting samples which exhibit anomalous behavior with respect to that observed from the training data at such layers (Section 2). With respect to similar approaches based on analyzing different network layers [14, 15], our defense does not require generating adversarial examples during training, and it is thus less computationally demanding.

We evaluate our defense against an adaptive white-box attacker that is aware of the defense mechanism and tries to bypass it. To this end, we propose a novel gradient-based attack that accounts for the rejection mechanism and aims to craft adversarial examples that avoid it (Section 3).

It is worth remarking here that correctly evaluating a defense mechanism is a crucial point when proposing novel defenses against adversarial examples [2, 17]. The majority of previous work proposing defense methods against adversarial examples has only evaluated such defenses against previous attacks rather than against an ad hoc attack crafted specifically against the proposed defense (see, e.g., [15, 18, 19] and all the other re-evaluated defenses in [17, 20]). The problem with these black-box and gray-box evaluations in which the attack is essentially unaware of the defense mechanism is that they are overly optimistic. It has indeed been shown afterwards that such defenses can be easily bypassed by simple modifications to the attack algorithm [17, 20, 21]. For instance, many defenses have been found to perform *gradient obfuscation*, i.e., they learn functions which are harder to optimize for gradient-based attacks; however, they can be easily bypassed by constructing a smoother, differentiable approximation of their function, e.g., via learning a surrogate model [2, 6, 22–25] or replacing network layers which obfuscate gradients with smoother mappings [17, 20, 21]. In our case, an attack that is unaware of the defense mechanism may tend to craft adversarial examples in areas of the input space which are assigned to the rejection class; thus, such attacks, as well as previously proposed ones, may rarely bypass our defense. For this reason, we believe that our adaptive white-box attack, along with the security evaluation methodology adopted in this work, provides another significant contribution to the state of the art related to the problem of properly evaluating defenses against adversarial examples.

The security evaluation methodology advocated in [2, 17, 26], which we also adopt in this work, consists of evaluating the accuracy of the system against attacks crafted with an increasing amount of perturbation. The corresponding *security evaluation curve* [2] shows how gracefully the performance decreases while the attack increases in strength, up to the point where the defense reaches zero accuracy. This is another important phenomenon to be observed, since any defense against test-time evasion attacks *has to fail* when the perturbation is sufficiently large (or, even better, unbounded); in fact, in the unbounded case, the attacker can ideally replace the source sample with any other sample from another class [17]. If accuracy under attack does not reach zero for very large perturbations, then it may be that the attack algorithm fails to find a good optimum (i.e., a good adversarial example). This in turn means that we are probably providing an optimistic evaluation of the defense. As suggested in [17], the purpose of a security evaluation should not be to show which attacks the defense withstands to, but rather to show when the defense fails. If one shows that larger perturbations that may compromise the content of the input samples and its nature (i.e., its true label) are required to break the defense, then we can retain the defense mechanism to be sufficiently robust. Another relevant point is to show that such a *breakdown* point occurs at a larger perturbation than that exhibited by competing defenses, to show that the proposed defense is more robust than previously proposed ones.

The empirical evaluation reported in Section 4, using both MNIST handwritten digits and CIFAR10 images, provides consistent results with the aforementioned aspects. First, it shows that our adaptive white-box attack is able to break our defensive method at larger perturbations. Second, it shows that our method improves the performance of competing rejection mechanisms which only leverage the deep representation learned at the output network layer. We thus believe that our analysis unveils a promising way of defending against adversarial examples.

We conclude the paper by discussing related work (Section 5), the main contributions of this work, and its limitations, along with promising future research directions (Section 6).

2 Deep neural rejection

The underlying idea of our DNR method is to estimate the distribution of unperturbed training points at different network layers and reject anomalous samples that may be incurred at test time, including adversarial examples. The architecture of DNR is shown in Fig. 1.

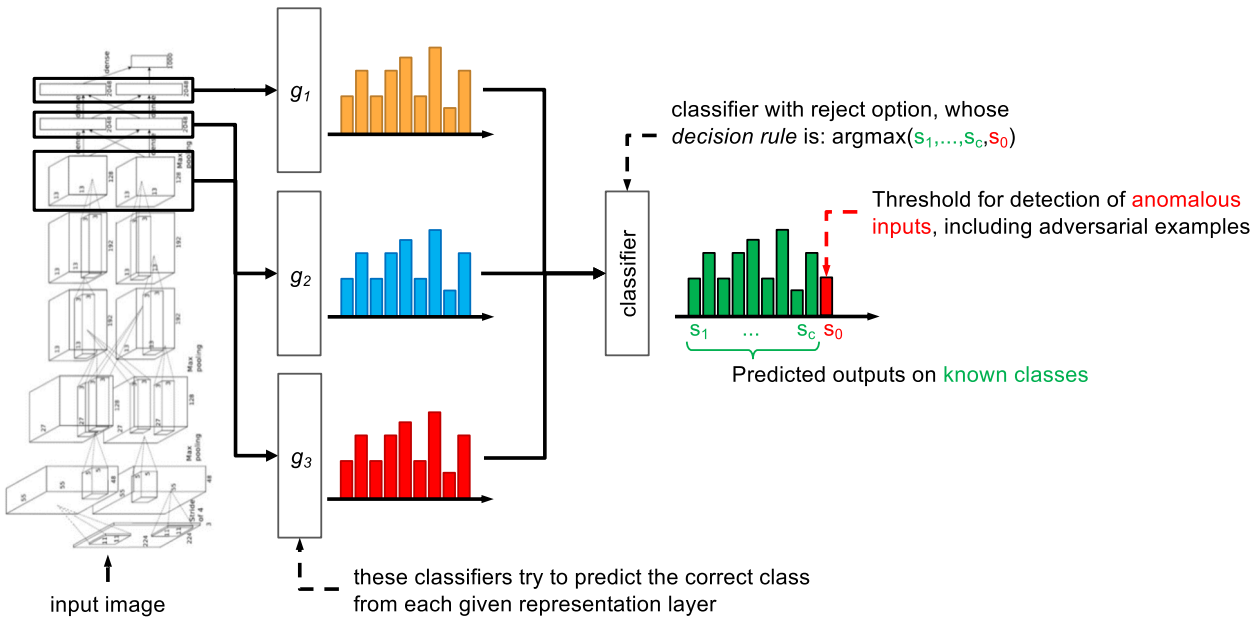

Fig. 1 Architecture of deep neural rejection (DNR). DNR considers different network layers and learns an SVM with the RBF kernel on each of their representations. The outputs of these SVMs are then combined using another RBF SVM, which will provide prediction scores $s_1, \ldots, s_c$ for each class. This classifier will reject samples if the maximum score $\max_{k=1,\ldots,c} s_k$ is not higher than the rejection threshold θ. This decision rule can be equivalently represented as $\arg\max_{k=0,\ldots,c} s_k(\boldsymbol{x})$, if we consider rejection as an additional class with $s_0 = \theta$

Before delving into the details of our method, let us introduce some notation. We denote the prediction function of a deep neural network with $f : \mathcal{X} \mapsto \mathcal{Y}$, where $\mathcal{X} \subseteq \mathbb{R}^d$ is the d-dimensional space of input samples (e.g., image pixels) and $\mathcal{Y} \subseteq \mathbb{R}^c$ is the space of the output predictions (i.e., the estimated confidence values for each class), being c the number of classes. If we assume that the network consists of m layers, then the prediction function f can be rewritten to make this explicit as $f(\phi_1(\phi_2(\ldots\phi_m(\boldsymbol{x};\boldsymbol{w}_m);\boldsymbol{w}_2);\boldsymbol{w}_1)$, where ϕ_1 and ϕ_m denote the mapping function learned respectively by the output and the input layer, and $\boldsymbol{w}_1$ and $\boldsymbol{w}_m$ are their weight parameters (learned during training).

For our defense mechanism to work, one has first to select a set of network layers, e.g., in Fig. 1, we select the outer layers ϕ_1, ϕ_2, and ϕ_3. Let us assume that the representation of the input sample $\boldsymbol{x}$ at level ϕ_i is $\boldsymbol{z}_i$. Then, on each of these selected representations, DNR learns an SVM with the RBF kernel $g_i(\boldsymbol{z}_i)$, trying to correctly predict the input sample. The confidence values on the c classes provided by this classifier are then concatenated with those provided by the other base SVMs and used to train a combiner, using again an RBF SVM.[1] The combiner will output predictions $s_1, \ldots, s_c$ for the c classes, but will reject samples if the maximum confidence score $\max_{k=1,\ldots,c} s_k$ is not higher than a rejection threshold θ. This decision rule can be compactly represented as $\arg\max_{k=0,\ldots,c} s_k(\boldsymbol{x})$, where we define an additional, *constant* output $s_0(\boldsymbol{x}) = \theta$ for the rejection class. According to this rule, if $s_0(\boldsymbol{x}) = \theta$ is the highest value in the set, the sample is rejected; otherwise, it is assigned to the class exhibiting the larger confidence value.

As proposed in [12], we use an RBF SVM here to ensure that the confidence values $s_1, \ldots, s_c$, for each given class, decrease while $\boldsymbol{x}$ moves further away from regions of the feature space which are densely populated by training samples of that class. This property, named *compact abating probability* in open-set problems [13, 28], is a desirable property to easily implement a *distance-based rejection* mechanism as the one required in our case to detect outlying samples. With respect to [12], we train this combiner on top of other base classifiers rather than only on the representation learned by the last network layer, to further improve the detection of adversarial examples. For this reason, in the following, we refer to the approach by Melis et al. [12], rejecting samples based only on their representation at the last layer, as *neural rejection* (NR), and to ours, exploiting also representations from other layers, as *deep neural rejection* (DNR).

3 Attacking deep neural rejection

To properly evaluate security, or *adversarial robustness*, of rejection-based defenses against adaptive white-box

[1] Validation samples should be used to train the combiner here and avoid overfitting, as suggested by stacked generalization [27].

adversarial examples, we propose the following. Given a source sample $\boldsymbol{x}$ and a maximum-allowed ε-sized perturbation, the attacker can optimize a defense-aware adversarial example $\boldsymbol{x}^\star$ by solving the following constrained optimization problem:

$$\boldsymbol{x}^\star = \arg \min_{\boldsymbol{x}':\|\boldsymbol{x}-\boldsymbol{x}'\|\leq\varepsilon} \Omega(\boldsymbol{x}') \text{ where } \quad \Omega(\boldsymbol{x}') = s_y(\boldsymbol{x}') - \max_{j\notin\{0,y\}} s_j(\boldsymbol{x}') \tag{1}$$

where $\|\boldsymbol{x} - \boldsymbol{x}'\| \leq \varepsilon$ 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 [29]), $y \in \mathcal{Y} = \{1,\ldots,c\}$ is the true class, and 0 is the rejection class. In practice, the attacker minimizes the output of the true class, while maximizing the output of the competing class (excluding the reject class) to achieve (untargeted) evasion. This amounts to performing a strong maximum-confidence evasion attack (rather than searching for a minimum-distance adversarial example). We refer the reader to [2, 6, 24] for a more detailed discussion on such topic. While we focus here on untargeted (error-generic) attacks, our formulation can be extended to account for targeted (error-specific) evasion as also done in [12].

The optimization problem in Eq. (1) can be solved through a standard projected gradient descent (PGD) algorithm, as given in Algorithm 1. In our experiments, we consider a variable step size η (by doubling the initial step size for ten times) and select the point $\boldsymbol{x}'$ minimizing the objective at each update step. This allows our attack to escape local minima which may hinder the optimization process, and consequently, it allows us to obtain a more reliable security evaluation of the proposed detection method.

Algorithm 1 PGD-based Maximum-confidence Adversarial Examples

Input: $\boldsymbol{x}_0$: the input sample; η: the 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.
Output: $\boldsymbol{x}'$: the adversarial example.

1: $\boldsymbol{x}' \leftarrow \boldsymbol{x}_0$
2: **repeat**
3: $\quad \boldsymbol{x} \leftarrow \boldsymbol{x}'$
4: $\quad \boldsymbol{x}' \leftarrow \Pi(\boldsymbol{x} - \eta\nabla\Omega(\boldsymbol{x}))$
5: **until** $|\Omega(\boldsymbol{x}') - \Omega(\boldsymbol{x})| \leq t$
6: **return** $\boldsymbol{x}'$

In Fig. 2, we report an example on a bi-dimensional toy problem to show how our defense-aware attack works against a rejection-based defense mechanism.

4 Experimental analysis

In this section, we evaluate the security of the proposed DNR method against adaptive, defense-aware adversarial examples. We consider two common computer vision benchmarks for this task, i.e., handwritten digit recognition (MNIST data) and image classification (CIFAR10 data). Our goal is to investigate whether and to which extent DNR can improve security against adversarial examples, in particular, compared to the previously proposed neural rejection (NR) defense (which only leverages the feature representation learned at the last network layer to reject adversarial examples) [12]. All the experiments presented in this section are based on the open-source Python library `secml` [30], which we plan to extend in the near future to include an implementation of both DNR and NR.

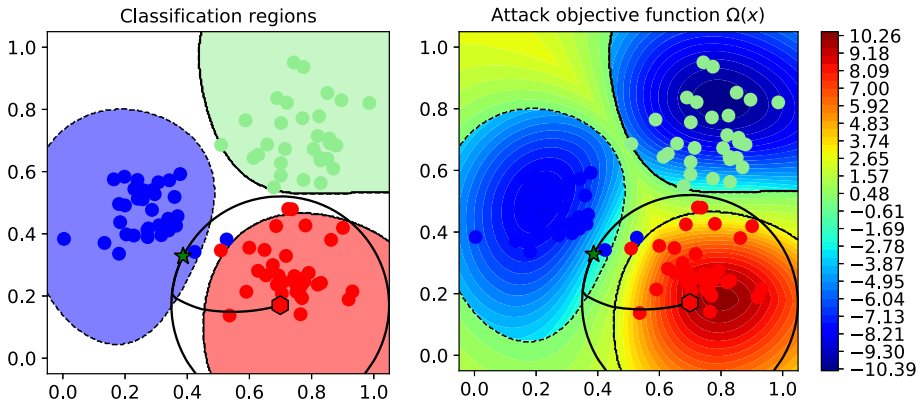

Fig. 2 Our defense-aware attack against an RBF SVM with rejection, on a 3-class bi-dimensional classification problem. The initial sample x_0 and the adversarial example $\mathbf{x}^\star$ are respectively represented as a red hexagon and a green star, while the ℓ_2-norm perturbation constraint $\|\mathbf{x}_0 - \mathbf{x}'\|_2 \leq \varepsilon$ is shown as a black circle. The left plot shows the decision region of each class, along with the reject region (in white). The right plot shows the values of the attack objective $\Omega(\boldsymbol{x})$ (in colors), which correctly enforces our attacks to avoid the reject region

Table 1 Model architecture for MNIST (left) and CIFAR10 (right) networks

Layer type	Dimension	Layer type	Dimension
Conv. + ReLU	32 filters (3 × 3)	Conv. + Batch Norm. + ReLU	64 filters (3 × 3)
Conv. + ReLU	32 filters (3 × 3)	Conv. + Batch Norm. + ReLU	64 filters (3 × 3)
Max Pooling	2 × 2	Max Pooling + Dropout ($p = 0.1$)	2 × 2
Conv. + ReLU	64 filters (3 × 3)	Conv. + Batch Norm. + ReLU	128 filters (3 × 3)
Conv. + ReLU	64 filters (3 × 3)	Conv. + Batch Norm. + ReLU	128 filters (3 × 3)
Max pooling	2 × 2	Max Pooling + Dropout ($p = 0.2$)	2 × 2
Fully connected + ReLU	200 units	Conv. + Batch Norm. + ReLU	256 filters (3 × 3)
Fully connected + ReLU	200 units	Conv. + Batch Norm. + ReLU	256 filters (3 × 3)
Softmax	10 units	Max Pooling + Dropout ($p = 0.3$)	2 × 2
		Conv. + Batch Norm. + ReLU	512 filters (3 × 3)
		Max Pooling + Dropout ($p = 0.4$)	2 × 2
		Fully Connected	512 units
		Softmax	10 units

4.1 Experimental setup

We discuss here the experimental setup used to evaluate our defense mechanism.

4.1.1 Datasets

As mentioned before, we run experiments on MNIST and CIFAR10 data. MNIST handwritten digit data consists of 60,000 training and 10,000 test gray-scale 28 × 28 images. CIFAR10 consists of 50,000 training and 10,000 test RGB 32×32 images. We normalized the images of both datasets in [0, 1] by simply dividing the input pixel values by 255.

4.1.2 Train-test splits

We average the results on five different runs. In each run, we consider 10,000 training samples and 1000 test samples, randomly drawn from the corresponding datasets. To avoid overfitting, we train the DNR combiner on the outputs of the base SVMs computed on a separate validation set, using a procedure known as *stacked generalization* [27]. We use a 3-fold cross-validation procedure to subdivide the training dataset into three folds. For three times, we learn the base SVMs on two folds and classify the remaining (validation) fold. We then concatenate the predicted values for each validation fold and use such values to train the combiner. The deep neural networks (DNNs) used in our experiments are pre-trained on a training dataset (different from the ones that we use to train the SVMs) of 30,000 and 40,000 training samples, respectively, for MNIST and CIFAR10.

4.1.3 Classifiers

We compare the DNR approach (which implements rejection here based on the representations learned by three different network layers) against an undefended DNN (without any rejection mechanism) and against the NR defense by Melis et al. [12] (which implements rejection on top of the representation learned by the output network layer). To implement the undefended DNNs for the MNIST dataset, we used the same architecture suggested by Carlini et al. [21]. For CIFAR10, instead, we considered a lightweight network that, despite its size, allows obtaining high performances. The two considered architectures are shown in Table 1, whereas Table 2 shows the model parameters that we used to train the overmentioned architectures. The three layers considered by the DNR classifier are the last three layers for the network trained on MNIST, and the last layer plus the last batch norm layer and the second to the last max-pooling layer for the one trained on CIFAR10 (chosen to obtain a reasonable amount of features).

4.1.4 Security evaluation

We compare these classifiers in terms of their security evaluation curves [2], reporting classification accuracy against an increasing ℓ_2-norm perturbation size ε, used to perturb all test samples. In particular, classification accuracy is computed as follows:

Table 2 Parameters used to train MNIST and CIFAR10 networks

Parameter	MNIST model	CIFAR model
Learning rate	0.1	0.01
Momentum	0.9	0.9
Dropout	0.5	(see Table 1)
Batch size	128	100
Epochs	50	75

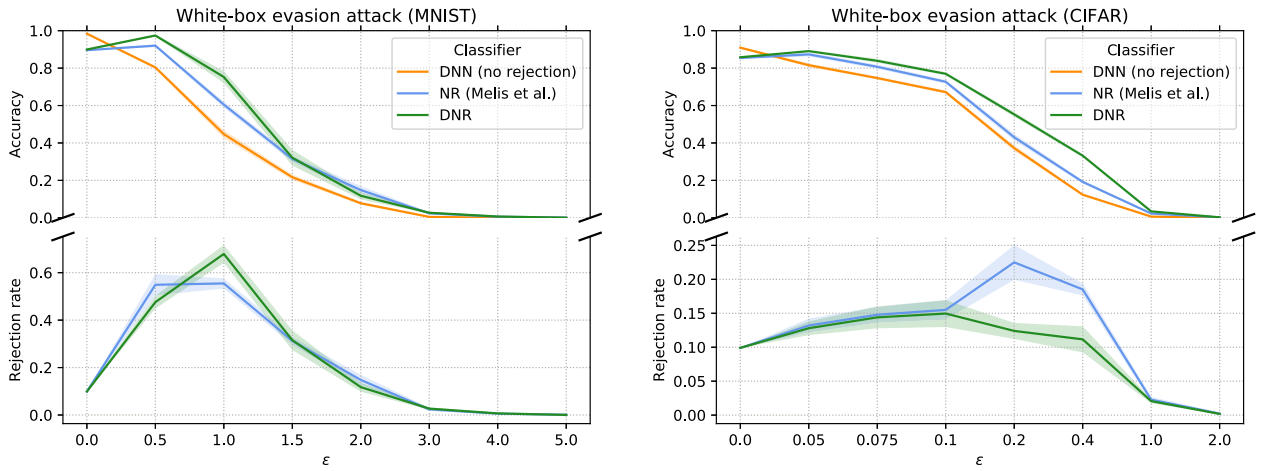

Fig. 3 Security evaluation curves for MNIST (left) and CIFAR10 (right) data, reporting mean accuracy (and standard deviation) against ε-sized attacks

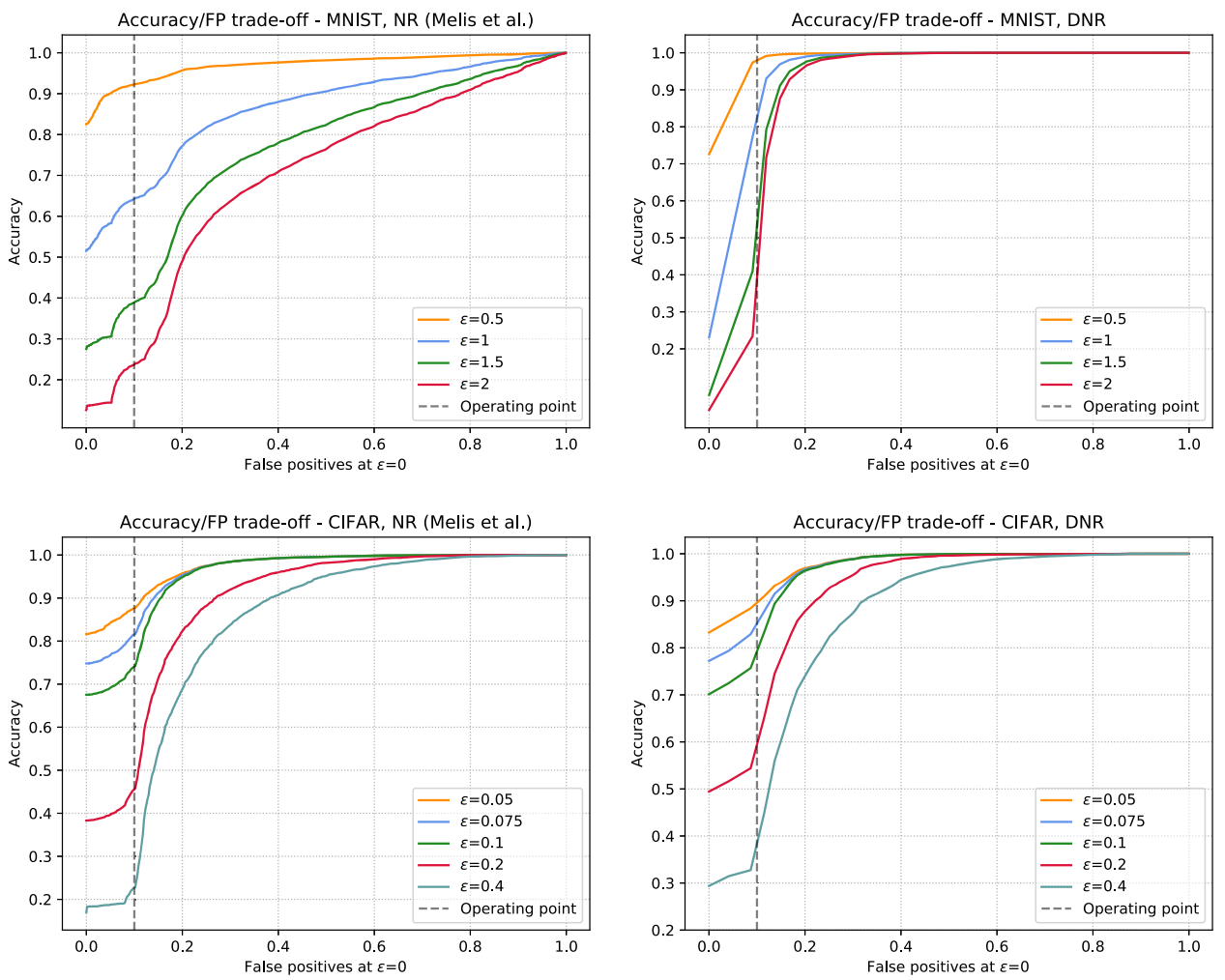

Fig. 4 Influence of the rejection threshold ? on classifier accuracy under attack (y-axis) vs false rejection rate (i.e., fraction of wrongly rejected unperturbed samples) on MNIST (top) and CIFAR10 (bottom) for NR (left) and DNR (right), for different ?-sized attacks. The dashed line highlights the performance at 10% false rejection rate (i.e., the operating point used in our experiments)

- In the absence of adversarial perturbation (i.e., for $\varepsilon = 0$), classification accuracy is computed as usual, but considering rejects as errors;
- In the presence of adversarial perturbation (i.e., for $\varepsilon > 0$), all test samples become adversarial examples, and we consider them correctly classified if they are assigned either to the rejection class or to their original class (which typically happens when the perturbation is too small to cause a misclassification).

For DNR and NR, we also report the rejection rates, computed by dividing the number of rejected samples by the number of test samples. Note that the difference between accuracy and rejection rate at each $\varepsilon > 0$ corresponds to the fraction of adversarial examples which are not rejected but still correctly assigned to their original class. Accordingly, under this setting, classifiers exhibiting higher accuracies under attack ($\varepsilon > 0$) can be retained more robust.

4.1.5 Parameter setting

We use a 5-fold cross-validation procedure to select the hyperparameters that maximize classification accuracy on the unperturbed training data, and set the rejection threshold θ for NR and DNR to reject 10% of the samples when no attack is performed (at $\varepsilon = 0$) .

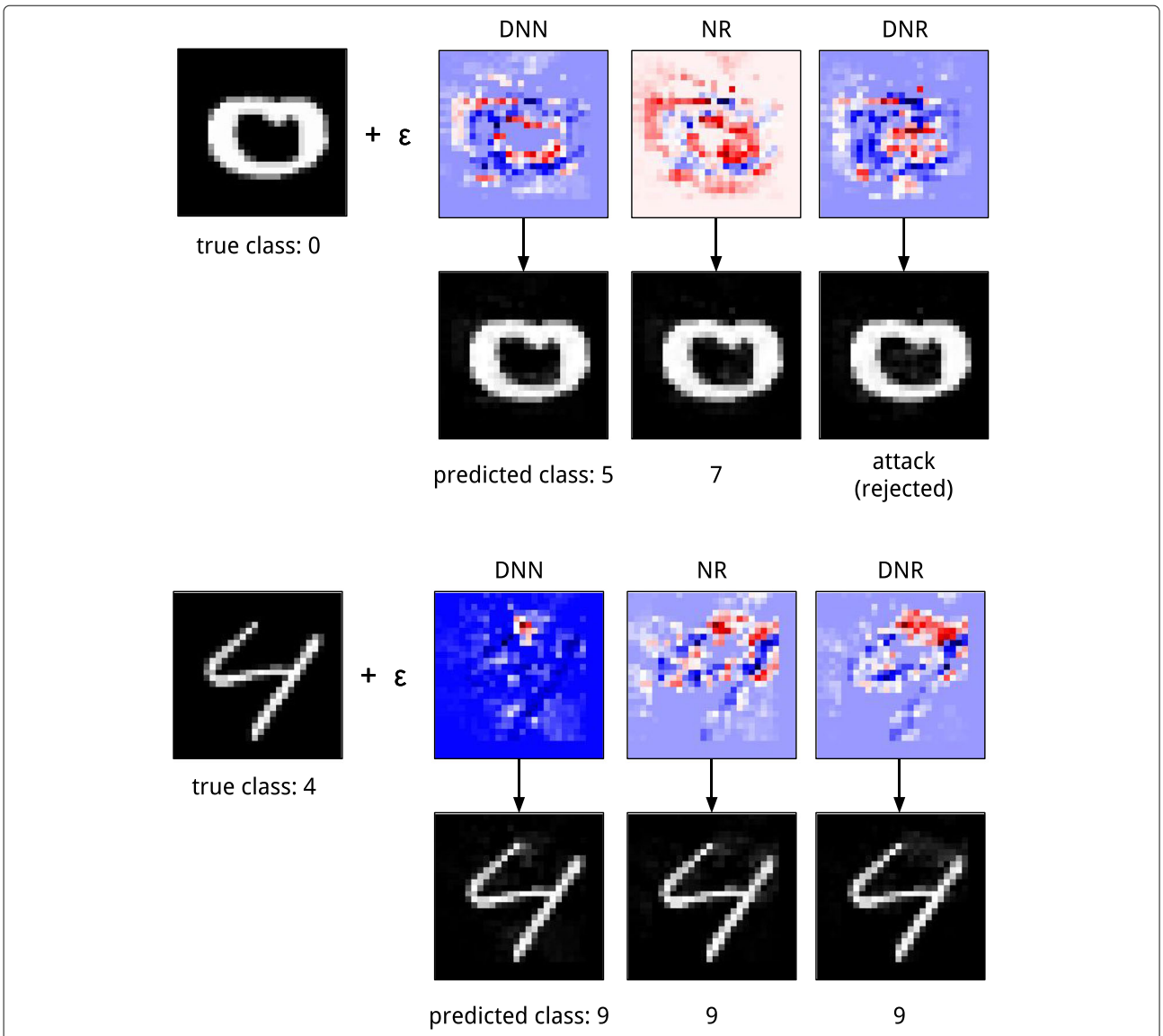

Fig. 5 Adversarial examples computed on the MNIST data to evade DNN, NR, and DNR. The source image is reported on the left, followed by the (magnified) adversarial perturbation crafted with $\varepsilon = 1$ against each classifier, and the resulting adversarial examples. We remind the reader that the attacks considered in this work are untargeted, i.e., they succeed when the attack sample is not correctly assigned to its true class

4.2 Experimental results

The results are reported in Fig. 3. In the absence of attack ($\varepsilon = 0$), the undefended DNNs slightly outperform NR and DNR, since the latter wrongly reject also some unperturbed samples.

Under attack, ($\varepsilon > 0$), when the amount of injected perturbation is exiguous, the rejection rate of both NR and DNR increases jointly with ε, as the adversarial examples are located far from the rest of the training classes in the representation space (i.e., the intermediate representations learned by the neural network). For larger ε, both NR and DNR can no longer correctly detect the adversarial examples, as they tend to become indistinguishable from the rest of the training samples (in the representation space in which NR and DNR operate). Both defenses outperform the undefended DNNs on the adversarial samples, and DNR slightly outperforms NR, exhibiting a more graceful decrease in performance. Although NR tends to reject more samples for $\varepsilon \in [0.1, 1]$ on CIFAR and for $\varepsilon = 0.5$ on MNIST, its accuracy is lower than DNR. The reason is that DNR remains more accurate than NR when classifying samples that are not rejected. This also means that DNR provides tighter boundaries closer to the training classes than NR, thanks to the exploitation of lower-level network representations, which makes the corresponding defended classifier more difficult to evade.

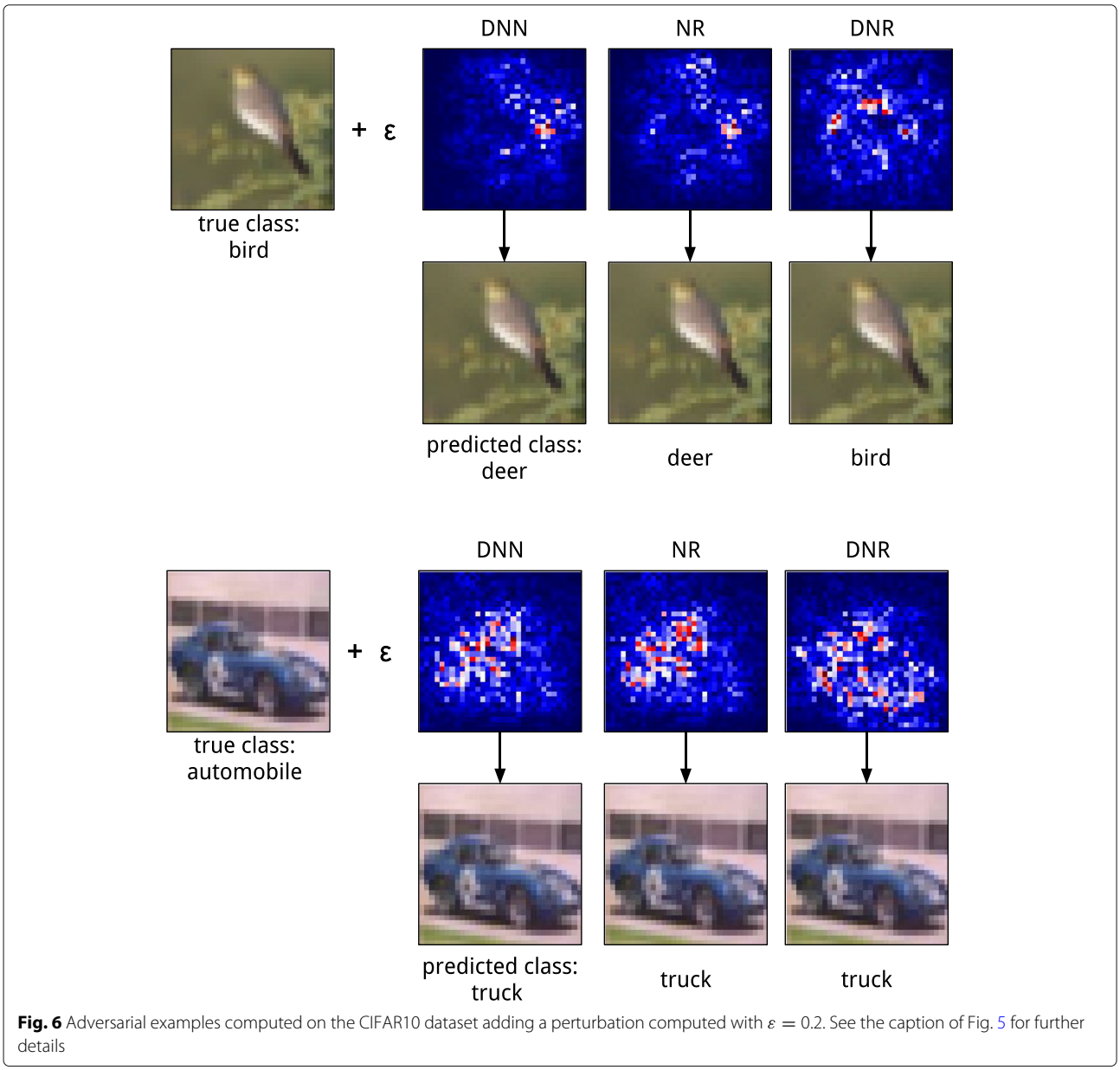

Fig. 6 Adversarial examples computed on the CIFAR10 dataset adding a perturbation computed with $\varepsilon = 0.2$. See the caption of Fig. 5 for further details

In Figs. 4 and 5, we show some adversarial examples computed respectively on the MNIST and CIFAR10 datasets against the considered classifiers.

Finally, in Fig. 6, we show how the selection of the rejection threshold θ allows us to trade security against adversarial examples (i.e., accuracy on the y-axis) for a more accurate classifier on the unperturbed samples (reported in terms of the rejection rate of unperturbed samples on the x-axis). In particular, increasing (decreasing) the rejection threshold amounts to increasing (decreasing) the fraction of correcly detected adversarial examples and to increasing (decreasing) the rejection rate when no attack is performed.

5 Related work

Different approaches have been recently proposed to perform rejection of samples that are outside of the training data distribution [31–33]. For example, Thulasidasan et al. [31] and Geifman et al. [32] have proposed novel loss functions accounting for rejection of inputs on which the classifier is not sufficiently confident. Geifman et al. [33] have proposed a method that allows the system designer to set the desired risk level by adding a rejection mechanism to a pre-trained neural network architecture. These approaches have, however, not been originally tested against adversarial examples, and it is thus of interest to assess their performance under attack in future work, also in comparison to our proposal.

Even if the majority of approaches implementing rejection or abstaining classifiers have not considered the problem of defending against adversarial examples, some recent work has explored this direction too [12, 13]. Nevertheless, with respect to the approach proposed in this work, they have only considered the output of the last network layer and perform rejection based solely on that specific feature representation. In particular, Bendale and Boult [13] have proposed a rejection mechanism based on reducing the open-set risk in the feature space of the activation vectors extracted from the last layer of the network, while Melis et al. [12] have applied a threshold on the output of an RBF SVM classifier. Despite these differences, the rationale of the two approaches is quite similar and resembles the older idea of distance-based rejection.

Few approaches have considered a multi-layer detection scheme similar to that envisioned in our work [14–16, 34, 35]. However, most of these approaches require generating adversarial examples at training time, which is computationally intensive, especially for high-dimensional problems and large datasets [14, 15, 34, 35]. Finding a methodology to tune the hyperparameters for generating the attack samples is also an open research challenge. Finally, even though the deep k-nearest neighbors approach by Papernot et al. [16] does not require generating adversarial examples at training time, it requires computing the distance of each test sample against all the training points at different network layer representations, which again raises scalability issues to high-dimensional problems and large datasets.

6 Conclusions and future work

We have proposed *deep neural rejection* (DNR), i.e., a multi-layer rejection mechanism, that, differently from other state-of-the-art rejection approaches against adversarial examples, does not require generating adversarial examples at training time, and it is less computationally demanding. Our approach can be applied to pre-trained network architectures to implement a defense against adversarial attacks at test time. The base classifiers and the combiner used in our DNR approach are trained separately. As future work, it would be interesting to perform an end-to-end training of the proposed classifier similarly to the approaches proposed in [31] and [32]. Another research direction may be that of testing our defense against training-time poisoning attacks [2, 5, 36–38].

Abbreviations

DNN: Deep neural network; NR: Neural rejection; DNR: Deep neural rejection; SVM: Support vector machine; RBF: Radial basis function

Authors' contributions

The authors read and approved the final manuscript.

Funding

This work was partly supported by the PRIN 2017 project RexLearn, funded by the Italian Ministry of Education, University and Research , and by the EU H2020 project ALOHA, under the European Union's Horizon 2020 research and innovation programme (grant no. 780788).

Availability of data and materials

The dataset MNIST and CIFAR10 analysed in this work are available respectively at the following web pages: http://yann.lecun.com/exdb/mnist/ and https://www.cs.toronto.edu/~kriz/cifar.html [39, 40].

Competing interests

The authors declare that they have no competing interests.

Author details

[1]DIEE, University of Cagliari, Piazza d'Armi, 09123 Cagliari, Italy. [2]Pluribus One, Cagliari, Italy. [3]Northwestern Polytechnical University, Xi'an, China.

Received: 30 September 2019 Accepted: 23 March 2020
Published online: 07 April 2020

References

1. A. D. Joseph, B. Nelson, B. I. P. Rubinstein, J. Tygar, *Adversarial machine learning*. (Cambridge University Press, 2018)
2. B. Biggio, F. Roli, Wild patterns: ten years after the rise of adversarial machine learning. Pattern. Recog. **84**, 317–331 (2018)
3. N. Dalvi, P. Domingos, Mausam, S. Sanghai, D. Verma, in *Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*. Adversarial classification, (Seattle, 2004), pp. 99–108
4. D. Lowd, C. Meek, in *Second Conference on Email and Anti-Spam (CEAS)*. Good word attacks on statistical spam filters (Mountain View, USA, 2005)
5. B. Biggio, B. Nelson, P. Laskov, in *29th Int'l Conf. on Machine Learning*, ed. by J. Langford, J. Pineau. Poisoning attacks against support vector machines (Omnipress, 2012), pp. 1807–1814

6. B. Biggio, I. Corona, D. Maiorca, B. Nelson, Šrndić, P. Laskov, G. Giacinto, F. Roli, in *Machine Learning and Knowledge Discovery in Databases (ECML PKDD), Part III*, ed. by H. Blockeel, K. Kersting, S. Nijssen, and F. Železný. Evasion attacks against machine learning at test time, LNCS, vol. 8190 (Springer Berlin Heidelberg, 2013), pp. 387–402
7. C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, R. Fergus, in *International Conference on Learning Representations*. Intriguing properties of neural networks (ICLR, Calgary, 2014)
8. I. J. Goodfellow, J. Shlens, C. Szegedy, in *International Conference on Learning Representations*. Explaining and harnessing adversarial examples (ICLR, San Diego, 2015)
9. A. Globerson, S. T. Roweis, in *Proceedings of the 23rd International Conference on Machine Learning*, ed. by W. W. Cohen, A. Moore. Nightmare at test time: robust learning by feature deletion, vol. 148 (ACM, 2006), pp. 353–360. https://doi.org/10.1145/1143844.1143889
10. M. Brückner, C. Kanzow, T. Scheffer, Static prediction games for adversarial learning problems. J. Mach. Learn. Res. 13, 2617–2654 (2012).
11. S. Rota Bulò, B. Biggio, I. Pillai, M. Pelillo, F. Roli, Randomized prediction games for adversarial machine learning. IEEE Trans. Neural Netw. Learn. Syst. 28(11), 2466–2478 (2017).
12. M. Melis, A. Demontis, B. Biggio, G. Brown, G. Fumera, F. Roli, in *ICCVW Vision in Practice on Autonomous Robots (ViPAR)*. Is deep learning safe for robot vision? Adversarial examples against the iCub humanoid (IEEE, 2017), pp. 751–759. https://doi.org/10.1109/iccvw.2017.94
13. A. Bendale, T. E. Boult, in *IEEE Conference on Computer Vision and Pattern Recognition*. Towards open set deep networks, (2016), pp. 1563–1572. https://doi.org/10.1109/cvpr.2016.173
14. F. Crecchi, D. Bacciu, B. Biggio, in *ESANN '19*. Detecting adversarial examples through nonlinear dimensionality reduction. In press
15. J. Lu, T. Issaranon, D. Forsyth, in *The IEEE International Conference on Computer Vision (ICCV)*. Safetynet: detecting and rejecting adversarial examples robustly, (2017). https://doi.org/10.1109/iccv.2017.56
16. N. Papernot, P. D. McDaniel, Deep k-nearest neighbors: Towards confident, interpretable and robust deep learning. CoRR. abs/1803.04765 (2018).
17. A. Athalye, N. Carlini, D. A. Wagner, in *ICML, vol. 80 of JMLR Workshop and Conference Proceedings*. Obfuscated gradients give a false sense of security: circumventing defenses to adversarial examples (JMLR.org, 2018), pp. 274–283
18. N. Papernot, P. McDaniel, X. Wu, S. Jha, A. Swami, in *2016 IEEE Symposium on Security and Privacy (SP)*. Distillation as a defense to adversarial perturbations against deep neural networks, (2016), pp. 582–597. https://doi.org/10.1109/sp.2016.41
19. D. Meng, H. Chen, in *24th ACM Conf. Computer and Comm. Sec. (CCS)*. MagNet: a two-pronged defense against adversarial examples, (2017). https://doi.org/10.1145/3133956.3134057
20. N. Carlini, D. A. Wagner, in *10th ACM Workshop on Artificial Intelligence and Security*, ed. by B. M. Thuraisingham, B. Biggio, D. M. Freeman, B. Miller, and A. Sinha. Adversarial examples are not easily detected: bypassing ten detection methods, AISec '17 (ACM, New York, 2017), pp. 3–14
21. N. Carlini, D. A. Wagner, in *IEEE Symposium on Security and Privacy*. Towards evaluating the robustness of neural networks (IEEE Computer Society, 2017), pp. 39–57. https://doi.org/10.1109/sp.2017.49
22. P. Russu, A. Demontis, B. Biggio, G. Fumera, F. Roli, in *9th ACM Workshop on Artificial Intelligence and Security*. Secure kernel machines against evasion attacks, AISec '16 (ACM, New York, 2016), pp. 59–69
23. N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, A. Swami, in *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*. Practical black-box attacks against machine learning, ASIA CCS '17 (ACM, New York, 2017), pp. 506–519
24. A. Demontis, M. Melis, M. Pintor, M. Jagielski, B. Biggio, A. Oprea, C. Nita-Rotaru, F. Roli, in *28th USENIX Security Symposium (USENIX Security 19)*. Why do adversarial attacks transfer? Explaining transferability of evasion and poisoning attacks (USENIX Association, 2019)
25. M. Melis, D. Maiorca, B. Biggio, G. Giacinto, F. Roli, in *26th European Signal Processing Conf.* Explaining black-box android malware detection, EUSIPCO (IEEE, Rome, 2018), pp. 524–528
26. B. Biggio, G. Fumera, F. Roli, Security evaluation of pattern classifiers under attack. IEEE Trans. Knowl. Data Eng. 26, 984–996 (2014).
27. D. H. Wolpert, Stacked generalization. Neural Netw. **5**, 241–259 (1992)
28. W. Scheirer, L. Jain, T. Boult, Probability models for open set recognition. IEEE Trans. Patt. An. Mach. Intell. 36(11), 2317–2324 (2014).
29. J. Duchi, S. Shalev-Shwartz, Y. Singer, T. Chandra, in *Proceedings of the 25th International Conference on Machine Learning, ICML '08*. Efficient projections onto the l1-ball for learning in high dimensions (ACM, New York, 2008), pp. 272–279
30. M. Melis, A. Demontis, M. Pintor, A. Sotgiu, B. Biggio, secml: A Python library for secure and explainable machine learning. arXiv (2019)
31. S. Thulasidasan, T. Bhattacharya, J. Bilmes, G. Chennupati, J. Mohd-Yusof, *Knows when it doesn't know: deep abstaining classifiers*, (OpenReview.net, 2019). https://openreview.net/forum?id=rJxF73R9tX
32. R. E.-Y. Yonatan Geifman, in *Proceedings of the 36th International Conference on Machine Learning, (ICML) 2019*. Selectivenet: a deep neural network with an integrated reject option, vol. 97 (PMLR, Long Beach, 2019), pp. 2151–2159
33. Y. Geifman, R. El-Yaniv, in *Advances in Neural Information Processing Systems 30*, ed. by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Selective classification for deep neural networks (Curran Associates, Inc., 2017), pp. 4878–4887
34. F. Carrara, R. Becarelli, R. Caldelli, F. Falchi, G. Amato, in *The European Conference on Computer Vision (ECCV) Workshops*. Adversarial examples detection in features distance spaces, (2018). https://doi.org/10.1007/978-3-030-11012-3_26
35. T. Pang, C. Du, J. Zhu, in *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*. Max-mahalanobis linear discriminant analysis networks (PMLR, 2018), pp. 4013–4022
36. M. Jagielski, A. Oprea, B. Biggio, C. Liu, C. Nita-Rotaru, B. Li, in *IEEE Symposium on Security and Privacy, SP '18*. Manipulating machine learning: poisoning attacks and countermeasures for regression learning (IEEE CS, 2018), pp. 931–947. https://doi.org/10.1109/sp.2018.00057
37. H. Xiao, B. Biggio, G. Brown, G. Fumera, C. Eckert, F. Roli, in *JMLR W&CP - Proc. 32nd Int'l Conf. Mach. Learning (ICML)*, ed. by F. Bach, D. Blei. Is feature selection secure against training data poisoning? vol. 37 (PMLR, Lille, 2015), pp. 1689–1698
38. S. Mei, X. Zhu, in *29th AAAI Conf. Artificial Intelligence (AAAI '15)*. Using machine teaching to identify optimal training-set attacks on machine learners, (Austin, 2015)
39. Y. Lecun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition. Proc. IEEE. 86, 2278–2324 (1998).
40. A. Krizhevsky, *Learning multiple layers of features from tiny images*. (University of Toronto, 2012)

Publisher's Note