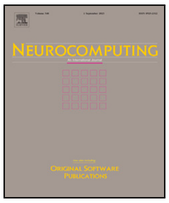
Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom



Unity is strength: Improving the detection of adversarial examples with ensemble approaches

Francesco Craighero [a], Fabrizio Angaroni [a,b], Fabio Stella [a,c], Chiara Damiani [d,e], Marco Antoniotti [a,c], Alex Graudenzi [a,c,f,*]

[a] Dept. of Informatics, Systems and Communication, Università degli Studi di Milano-Bicocca, Viale Sarca 336, Milan, 20126, Italy
[b] Computational Biology Research Centre, Human Technopole, Viale Rita Levi-Montalcini 1, Milan, 20157, Italy
[c] B4 - Bicocca Bioinformatics Biostatistics and Bioimaging Centre, Università degli Studi di Milano-Bicocca, Via Follereau 3, Vedano al Lambro (MB), 20854, Italy
[d] Dept. of Biotechnology and Biosciences, Università degli Studi di Milano-Bicocca, Piazza della Scienza, 2, Milan, 20126, Italy
[e] SYSBIO/ISBE.IT Centre of Systems Biology, Piazza della Scienza, 2, Milan, 20126, Italy
[f] Institute of Molecular Bioimaging and Physiology, National Research Council (IBFM-CNR), Via Fratelli Cervi 93, Segrate (MI), 20054, Italy

ARTICLE INFO

Communicated by N. Zeng

Dataset link: https://github.com/BIMIB-DISCo/ENAD-experiments

MSC:
68T07
68T45

Keywords:
Computer vision
Deep learning
Adversarial example detection
Ensemble
One-class support vector machines

ABSTRACT

A key challenge in computer vision and deep learning is the definition of robust strategies for the detection of adversarial examples. In this work, we propose the adoption of ensemble approaches to leverage the effectiveness of multiple detectors in exploiting distinct properties of the input data. To this end, the ENsemble Adversarial Detector (ENAD) framework integrates scoring functions from state-of-the-art detectors based on Mahalanobis distance, Local Intrinsic Dimensionality, and One-Class Support Vector Machines, which process the hidden features of deep neural networks. ENAD is designed to ensure high standardization and reproducibility to the computational workflow.

Extensive tests on benchmark datasets, models and adversarial attacks show that ENAD outperforms all competing methods in the large majority of settings. The improvement over the state-of-the-art and the intrinsic generality of the framework, which allows one to easily extend ENAD to include any set of detectors and integration strategies, set the foundations for the new area of ensemble adversarial detection.

Introduction

Deep Neural Networks (DNNs) have achieved impressive results in complex machine learning tasks, in a variety of fields such as computer vision [3] and computational biology [4].

However, recent studies have shown that state-of-the-art DNNs for object recognition tasks are vulnerable to *adversarial examples* [5,6]. For instance, in the field of computer vision, adversarial examples are perturbed images that are misclassified by a given DNN, even if being almost indistinguishable from the original (and correctly classified) image. Adversarial examples have been investigated in many additional real-world applications and settings, including malware detection [7] and speech recognition [8].

Thus, understanding and countering adversarial examples has become a crucial challenge for the widespread adoption of DNNs in safety-critical settings, and resulted in the development of an ever-growing number of *defensive techniques*. Among the possible countermeasures, some aim at increasing the robustness of the DNN model during the training phase, via *adversarial training* [5,6], *defensive distillation* [9] or by training more robust models [10–12] (sometimes referred to as *proactive* methods). Alternative approaches aim at *detecting* adversarial examples in the test phase, by defining specific functions for their detection and filtering-out (*reactive* methods).

Existing adversarial detection approaches (reviewed in Section 1.3) can be categorized depending on the features that they take into account: the input example itself, like ExAD [13] that characterizes adversarials by their feature attribution maps, the hidden features, such as the LID detector [2] that detects adversarials based on their local intrinsic dimensionality, and those that consider the output, such as the algorithm proposed in [14] that evaluates the distribution of the logits in the vicinity of a point to determine whether it is a clean or adversarial example. Detecting adversarials solely based on the input example is a challenging task, as adversarials are designed to be very similar to clean images. As a result, most state-of-the-art detectors rely on either

* Corresponding author at: Dept. of Informatics, Systems and Communication, Università degli Studi di Milano-Bicocca, Viale Sarca 336, Milan, 20126, Italy.
E-mail address: alex.graudenzi@unimib.it (A. Graudenzi).

https://doi.org/10.1016/j.neucom.2023.126576
Received 28 March 2023; Received in revised form 18 June 2023; Accepted 16 July 2023
Available online 23 July 2023

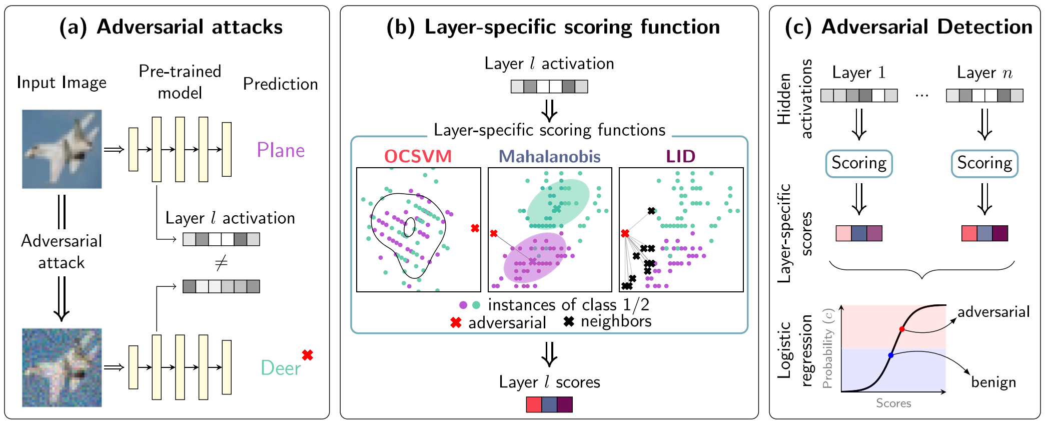


Fig. 1. ENAD framework for adversarial detection. A schematic depiction of the ENAD framework is displayed. **(a)** Given an input image, which can be either benign or adversarial, and a pre-trained deep neural network, the activations of the hidden layers are extracted. **(b)** In order to measure the distance of the image with respect to training examples, layer-specific scores are computed via functions based either on One-class Support Vector Machines, Mahalanobis distance [1] or Local Intrinsic Dimensionality [2]. **(c)** In the current implementation, layer- and detector-specific scores are integrated via logistic regression, so to classify the image as benign or adversarial, with confidence c.

the hidden features or the logits of a model to identify adversarials. However, as discussed in this study, using the hidden features can be challenging due to the high-dimensionality of the data, and selecting and combining the distinct layers requires careful consideration. Lastly, the most common approach for detecting adversarials from the logits is to analyze the neighborhood of normal examples, and in doing so, it is crucial to identify meaningful and effective sources of noise.

In this paper, we introduce a novel *ensemble approach* for the detection of adversarial examples, named ENsemble Adversarial Detector (ENAD), which integrates scoring functions computed from multiple detectors that process the hidden layer activations of pre-trained DNNs. The underlying rationale is that, given the high-dimensionality of the hidden layers of Convolutional Neural Networks (CNNs) and the difficulty of the adversarial detection problem, different algorithmic strategies might be effective in capturing and exploiting distinct properties of adversarial examples. Accordingly, their combination might improve the detection performance since ensembles are well-known methods to achieve high-performance classifiers [15], as also suggested in the similar topic of outlier detection [16]. To the best of our knowledge, this is the first time that an ensemble approach is applied to the hidden features of DNNs for adversarial detection, whereas ensemble-based proactive defences have been previously introduced [10–12] and, recently, two ensembling methods for adversarial [13] (although not being focused on latent features), and out-of-distribution detection [17] have been proposed. A comparison between ENAD and ExAD, introduced in [13], is available in Section 3.1.4.

In detail, ENAD includes two state-of-the-art detectors, based on Mahalanobis distance [1] and Local Intrinsic Dimensionality (LID) [2], and a newly developed detector based on One-Class SVMs (OCSVMs) [18]. OCSVMs were previously adopted for adversarial detection in [19], but we extended the previous method by defining both a pre-processing step and a Bayesian hyperparameter optimization strategy, both of which result fundamental to achieve performances comparable to [1,2]. We selected these specific adversarial detectors for two reasons. Firstly, all three detectors demonstrate state-of-the-art performance and employ the same type of features, which allows us to construct an ensemble. Secondly, each detector adopts a distinct approach: local intrinsic dimensionality, kernel and covariance estimation for LID, OCSVM, and Mahalanobis, respectively, which enables us to consider different perspectives on adversarial detection. In the current implementation, the output of each detector is then integrated via a logistic regression that returns both the adversarial classification and the overall confidence of the prediction. A schematic depiction of the ENAD framework is provided in Fig. 1.

For the sake of reproducibility, the performances of ENAD and competing methods were assessed with the extensive setting originally proposed in [1]. In particular, we performed experiments with two models, namely ResNet [20] and DenseNet [21], trained on CIFAR-10 [22], CIFAR-100 [22] and SVHN [23], and considered four benchmark adversarial attacks, i.e., FGSM [6], BIM [24], DeepFool [25] and CW [26]. In the various tests, ENAD was compared against its constituting standalone detectors (Section 3.1.2), other ensemble strategies (Section 3.1.3) and additional similar adversarial detectors (Section 3.1.4). We also executed an additional array of experiments aimed at assessing the performance of the distinct detectors when trained on a given attack and tested against others (*transfer attacks*).

Main contributions. The main contributions of the article can be summarized as follows.

- *Improvement of OCSVM performance in adversarial detection*: we introduced an evolved version of the OCSVM strategy for adversarial detection (first described in [19]), by designing a new pipeline with Bayesian hyperparameter optimization and data preprocessing on top of the default training process. Results highlight performance improvements.
- *Assessment of layer- and attack-specificity of standalone detectors*: thanks to extensive tests on benchmark datasets, models and attacks, we show that the performance of state-of-the-art standalone detectors is layer- and attack-specific, possibly guiding the improvements of such methods. Importantly, we demonstrate that the predictions of different detectors are scarcely overlapping.
- *Introduction of the ENAD ensemble framework for adversarial detection*: we propose a new detector, named ENAD, which integrates layer-specific scoring functions from multiple independent detectors. Its performance is assessed against standalone detectors, different ensemble strategies and other integration schemes (e.g., voting) in a variety of experimental settings. The framework is described by clearly stating all design choices, paving the way to future extensions with different detectors and integration schemes.
- *Performance evaluation in attack transfer settings*: we present a quantitative evaluation of the performance of ENAD and competing methods in transfer attack settings. We show the limitations of existing strategies and provide guidelines for future research.
- *Visualization of adversarial examples in low-dimensional space*: we deliver an easy-to-interpret way of visualizing adversarial examples on a low-dimensional projection of the score space, which provides a proxy of their "hardness", and may be particularly useful in real-world applications.

1. Background

1.1. Notation

Let us consider a multiclass classification problem with $C > 2$ classes. Let $\mathcal{N}(\boldsymbol{x}) = \sigma_{\text{lgt}} \circ h_L \circ \ldots \circ h_1(\boldsymbol{x})$ be a DNN with L layers, where h_l is the lth hidden layer and σ_{lgt} is the output layer, i.e. the C logits. To simplify the notation, we refer to the activation of the lth hidden layer given the input example $\boldsymbol{x}$, i.e. to $h_l \circ \ldots \circ h_1(\boldsymbol{x})$, as $h_l(\boldsymbol{x})$ and to the logits as $\sigma_{\text{lgt}}(\boldsymbol{x})$. Let $\sigma_{\text{sm}}(\boldsymbol{x}) = \sigma_{\text{sm}}(\sigma_{\text{lgt}}(\boldsymbol{x}))$ be the softmax of $\sigma_{\text{lgt}}(x)$, then the predicted class is $\hat{y} = \arg\max_k \sigma_{\text{sm}}(\boldsymbol{x})^k$ with confidence $\hat{p} = \max_k \sigma_{\text{sm}}(\boldsymbol{x})^k$. Lastly, by $J(\boldsymbol{x}, t) = -\log \sigma_{\text{sm}}(\boldsymbol{x})^t$ we will denote the cross-entropy loss function given input $\boldsymbol{x}$ and target class $t \in [1, C]$.

1.2. Adversarial attacks algorithms

In the following, we describe the adversarial attacks that were employed in our experiments, namely FGSM [6], BIM [24], DeepFool [25] and CW [27]. A schematic representation of the adversarial generation pipeline is available in Supplementary Fig. 1.

- FGSM [6] defines optimal L_∞ constrained perturbations as:

  $$\tilde{\boldsymbol{x}} = \boldsymbol{x} + \epsilon \cdot \text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{x}, t)),$$

  such that ϵ is the minimal perturbation in the direction of the gradient with respect to the input image ($\nabla_{\boldsymbol{x}}$) that changes the prediction of the model from the true class y to the target class t.
- BIM [24] extends FGSM by applying it k times with a fixed step size α, while also ensuring that each perturbation remains in the ϵ-neighborhood of the original image x by using a per-pixel clipping function clip:

  $$\tilde{\boldsymbol{x}}_{\boldsymbol{0}} = \boldsymbol{x}$$

  $$\tilde{\boldsymbol{x}}_{\boldsymbol{n+1}} = \text{clip}_{\boldsymbol{x},\epsilon}(\boldsymbol{x}_{\boldsymbol{n}} + \alpha \cdot \text{sign}(\nabla_{\tilde{\boldsymbol{x}}_{\boldsymbol{n}}} J(\tilde{\boldsymbol{x}}_{\boldsymbol{n}}, t)))$$

- DeepFool [25] iteratively finds the optimal L_2 perturbations that are sufficient to change the target class by approximating the original non-linear classifier with a linear one. Thanks to the linearization, in the binary classification setting the optimal perturbation corresponds to the distance to the (approximated) separating hyperplane, while in the multiclass the same idea is extended to a one-vs-all scheme. In practice, at each ith step the method computes the optimal perturbation p_i of the simplified problem, until $\tilde{\boldsymbol{x}} = \boldsymbol{x} + \sum_i p_i$ is misclassified.
- the CW attack [27], in two variants:
  - the original L_2 norm attack, that we will refer as CW: this variant uses gradient descent to minimize $\|\tilde{\boldsymbol{x}} - \boldsymbol{x}\|^2 + c \cdot l_{cw}(\tilde{\boldsymbol{x}})$, where the loss l_{cw} is defined as:

    $$l_{cw}(\boldsymbol{x}) = \max(\max\{\sigma_{\text{lgt}}(\tilde{\boldsymbol{x}})^i : i \neq t\} - \sigma_{\text{lgt}}(\tilde{\boldsymbol{x}})^t, -\kappa).$$

    The objective of the optimization is to minimize the L_2 norm of the perturbation and to maximize the difference between the target logit $\sigma_{\text{lgt}}(\tilde{\boldsymbol{x}})^t$ and the one of the next most likely class up to real-valued constant κ, that models the desired confidence of the crafted adversarial.
  - the L_∞ norm variant defined in [28], that we will refer to as CW$_\infty$, since it employs the same l_{cw} loss. In this variant, we optimize for l_{cw}, while clipping the adversarial such that $\|\tilde{\boldsymbol{x}} - \boldsymbol{x}\|^\infty = \epsilon$, with ϵ given as a parameter. Furthermore, we use the constant κ to obtain high-confidence adversarials.

1.3. Adversarial detection

Let us consider a classifier $\mathcal{N}$ trained on a training set $\mathcal{X}^{train}$ and a test example $\boldsymbol{x}_{\boldsymbol{0}} \in \mathcal{X}^{test}$, with predicted label $\hat{y}_0$. Adversarial examples detectors can be broadly categorized according to the features they consider: (i) the features of the test example $\boldsymbol{x}_{\boldsymbol{0}}$, ($ii$) the hidden features $h_l(\boldsymbol{x}_{\boldsymbol{0}})$, or ($iii$) the features of the output of the network $out(\boldsymbol{x}_{\boldsymbol{0}})$, i.e., the logits $\sigma_{\text{lgt}}(\boldsymbol{x}_{\boldsymbol{0}})$ or the confidence scores $\sigma_{\text{sm}}(\boldsymbol{x}_{\boldsymbol{0}})$.

The first family of detectors tries to distinguish normal from adversarial examples by focusing on the input data. For instance, in [29] the coefficients of low-ranked principal components of $\boldsymbol{x}_{\boldsymbol{0}}$ are used as features for the detector. In [30], the authors employed the statistical divergence, such as Maximum Mean Discrepancy, between $\mathcal{X}^{train}$ and $\mathcal{X}^{test}$ to detect the presence of adversarial examples in $\mathcal{X}^{test}$. Lastly, in [13] the feature attributions of the input image are considered as features of deep models for anomaly detection.

The second family of detectors aims at exploiting the information of the hidden features $h_l(\boldsymbol{x}_{\boldsymbol{0}})$. In this group, some detectors rely on the identification of the nearest neighbors of $h_l(\boldsymbol{x}_{\boldsymbol{0}})$ to detect adversarial examples, by considering either: the Euclidean distance to the neighbors [31], the conformity of the predicted class among the neighbors [32,33], the Local Intrinsic Dimensionality [2], the impact of the nearest neighbors on the classifier decision [34], or the prediction of a graph neural-network trained on the nearest neighbors graph [35]. In the same category, additional detectors take into account the conformity of the hidden representation $h_l(\boldsymbol{x}_{\boldsymbol{0}})$ to the hidden representation of instances with the same label in the training set, i.e., to $\{h_l(\boldsymbol{x}_{\boldsymbol{0}}) : \hat{y}_0 = y, (\boldsymbol{x}, y) \in X^{train}\}$, by computing either the Mahalanobis distance [1,36] to the class means, or the likelihood of a Gaussian Mixture Model (GMM) [37]. Other detectors take the hidden representation $h_l(\boldsymbol{x}_{\boldsymbol{0}})$ itself as a discriminating feature, by training either a DNN [38], a Support Vector Machine (SVM) [39], a One-class Support Vector Machine (OCSVM) [19] or by using a kernel density estimate [40]. Additional methods within this family use the hidden representation $h_l(\boldsymbol{x}_{\boldsymbol{0}})$ as a feature to train a predictive model m_l. The model m_l either seeks to predict the same C classes of the original classifier [19,41] or to reconstruct the input data $\boldsymbol{x}_{\boldsymbol{0}}$ from $h_l(\boldsymbol{x}_{\boldsymbol{0}})$ [29,42]. The detector then classifies $\boldsymbol{x}_{\boldsymbol{0}}$ as adversarial/benign either by relying on the confidence of the prediction $\hat{y}_0$ in the former case or on its reconstruction error in the latter.

The third family of detectors employs the output of the network $out(\boldsymbol{x}_{\boldsymbol{0}})$ to detect adversarial inputs. In this category, some detectors consider the divergence between $out(\boldsymbol{x}_{\boldsymbol{0}})$ and $out(\phi(\boldsymbol{x}_{\boldsymbol{0}}))$ where ϕ is a function such as a squeezing function that reduces the features of the input [43], an autoencoder trained on $\mathcal{X}^{train}$ [44], a denoising filter [45] or a random perturbation [14,46] , or an operation of erase and restore of random pixels [47]. Some others take the confidence score of the predicted class $\hat{y}$, which is expected to be lower when the example is anomalous [29,42,48]. Lastly, in [49] a DNN detector was trained directly on the logits, whereas in [40] Bayesian uncertainty of dropout DNNs was used as a feature for the detector.

Detectors exist that do not fall within any of the above families, which employ, for instance, the layer-wise norm of the gradients [50] and the consistency of the softmax scores $\sigma_{\text{sm}}(\boldsymbol{x}_{\boldsymbol{0}})$ of multiple models [51].

2. Methods

In this section, we illustrate the ENAD ensemble approach for adversarial detection, as well as the properties of the scoring functions it integrates, respectively based on OCSVM (Detector A - *new*), Mahalanobis (Detector B) and LID (Detector C), which can also be used as standalone detectors. We also describe the background of both adversarial examples generation and detection, the partitioning of the input data and the extraction of the features processed by ENAD and standalone detectors.

2.1. Data partitioning

Let $\mathcal{X}^{train}$ be the training set on which the classifier $\mathcal{N}$ was trained, $\mathcal{X}^{test}$ the test set and $\mathcal{L}_{norm} \subseteq \mathcal{X}^{test}$ the set of correctly classified test

Algorithm 1 OCSVM detector (see the main text for an explanation of the notation employed).

```
Input: Act. $h_l$ of layer $l$, trainset $\mathcal{X}^{train}$, labeled set $\mathcal{L}$
 1: for each $l$ in $1, \dots, L$ do
 2:     Centering and PCA-whitening of $h_l$: $h_l^*$
 3:     Select best layer-specific parameters $\theta = \{\nu, \gamma\}$
 4:     Fit OCSVM$_l(\theta)$ on $\{h_l^*(\boldsymbol{x}) : \boldsymbol{x} \in \mathcal{X}^{train}\}$
 5:     OCSVM$_l(\theta)$ decision function: $\mathrm{O}_l$
 6:     Layer $l$ score of $\boldsymbol{x_0}$: $\mathrm{O}_l(\boldsymbol{x_0})$
 7: end for
 8: Scores vector: $\mathbf{O}(\boldsymbol{x_0}) := [\mathrm{O}_1(\boldsymbol{x_0}), \dots, \mathrm{O}_L(\boldsymbol{x_0})]$
 9: Fit $adv$ posterior on $\mathcal{L}^{train}$: $p(adv|\mathbf{O}(\boldsymbol{x_0}))$
10: OCSVM of $\boldsymbol{x_0}$: OCSVM$(\boldsymbol{x_0}) := p(adv|\mathbf{O}(\boldsymbol{x_0}))$
11: return OCSVM
```

instances. Following the setup done in [1,2], from $\mathcal{L}_{norm}$ we generate (i) a set of noisy examples $\mathcal{L}_{noisy}$ by adding random Gaussian noise, with the additional constraint of being correctly classified, and (ii) a set of adversarial examples $\mathcal{L}_{adv}$ generated via a given attack. We also ensure that $\mathcal{L}_{norm}$, $\mathcal{L}_{noisy}$ and $\mathcal{L}_{adv}$ have the same size. The set $\mathcal{L} = \mathcal{L}_{norm} \cup \mathcal{L}_{noisy} \cup \mathcal{L}_{adv}$ will be our *labeled dataset*, where the label is adv for adversarial examples and $\overline{adv}$ for benign ones. As detailed in the following sections, $\mathcal{L}$ will be split into a training set $\mathcal{L}^{train}$, a validation set $\mathcal{L}^{valid}$ for hyperparameter tuning and a test set $\mathcal{L}^{test}$ for the final evaluation.

2.2. Feature extraction

In our experimental setting, $h_l(\boldsymbol{x})$, with $l \in [1, \dots, L]$, corresponds to either the first convolutional layer or to the output of the lth dense (residual) block of a DNN (e.g. DenseNet or ResNet). As proposed in [1], the size of the feature map is reduced via average pooling, so that $h_l(\boldsymbol{x})$ has a number of features equal to the number of channels of the lth layer. Detectors A, B, and C and ENAD are applied to such set of features, as detailed in the following.

2.3. Detector A: OCSVM

This newly designed detector is based on a standard anomaly detection technique called One-Class SVM (OCSVM) [18], which belongs to the family of one-class classifiers [52]. One-class classification is a problem in which the classifier aims at learning a good description of the training set and then rejects the inputs that do not resemble the data it was trained on, which represent outliers or anomalies. This kind of classifier is usually adopted when only one class is sufficiently represented within the training set, while the others are undersampled or hard to be characterized, as in the case of adversarial examples, or anomalies in general. OCSVM was first employed for adversarial detection in [19]. Here, we modified it by defining an input preprocessing step based on PCA-whitening [53], and by employing a Bayesian optimization technique [54] for hyperparameter tuning. The pseudocode is reported in Algorithm 1.

Preprocessing. OCSVM employs a kernel function (in our case a Gaussian RBF kernel) that computes the Euclidean distance among data points. Hence, it might be sound to standardize all the features of the data points, at the preprocessing stage, to make them equally important. To this end, each hidden layer activation $h_l(\boldsymbol{x_0})$ is first centered on the mean activations $\mu_{l,c}$ of the examples of the training set $\mathcal{X}^{train}$ of class c. Then, PCA-whitening $\mathbf{W}_l^{PCA}$ is applied:

$$\begin{aligned} h_l^*(\boldsymbol{x_0}) &= \mathbf{W}_l^{PCA} \cdot (h_l(\boldsymbol{x_0}) - \mu_{l,c}) \\ &= \boldsymbol{\Lambda}_l^{-1/2} \cdot \boldsymbol{U}_l^T \cdot (h_l(\boldsymbol{x_0}) - \mu_{l,c}), \end{aligned}$$

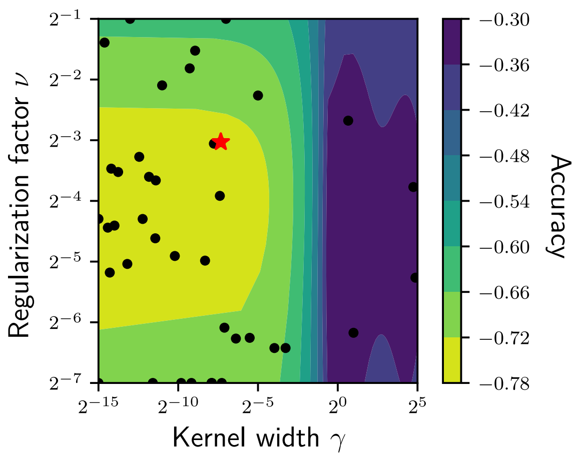


Fig. 2. OCSVM hyperparameter optimization. The influence of different combinations of OCSVM hyperparameters $\{\nu, \gamma\}$ on the validation accuracy is explored via Bayesian optimization [56], in the example scenario of DenseNet model, CIFAR-10 dataset and DeepFool attack. The gradient returns the validation accuracy estimated on $\mathcal{L}^{valid}$. The red star represents the optimal configuration, which is then employed for adversarial detection in both the OCSVM and the ENAD detectors.

where $\boldsymbol{U}_l^T$ is the eigenmatrix of the covariance $\boldsymbol{\Sigma}_l$ of activations h_l and $\boldsymbol{\Lambda}_l$ is the eigenvalues matrix of the examples of $\mathcal{X}^{train}$. Whitening is a commonly used preprocessing technique for outlier detection, since it enhances the separation of points that deviate in low-variance directions [55]. Moreover, in [36] it was conjectured that the effectiveness of the Mahalanobis distance [1] for out-of-distribution and adversarial detection is due to the strong contribution of low-variance directions. Thus, this preprocessing step allows the one-class classifier to achieve better overall performances[1].

Layer-specific scoring function. After preprocessing, OCSVM with a Gaussian RBF kernel is trained on the hidden layer activations h_l of layer l of the training set $\mathcal{X}^{train}$. Once the model has been fitted, for each instance $\boldsymbol{x_0}$ layer-specific scores $\mathbf{O}(\boldsymbol{x_0}) = [\mathrm{O}_1(\boldsymbol{x_0}), \mathrm{O}_2(\boldsymbol{x_0}), \dots, \mathrm{O}_L(\boldsymbol{x_0})]$ are evaluated. More in detail, let $\mathcal{S}_l$ be the set of support vectors, the decision function $\mathrm{O}_l(\boldsymbol{x_0})$ for the lth layer is computed as:

$$\mathrm{O}_l(\boldsymbol{x_0}) = \sum_{sv \in \mathcal{S}_l} \alpha_{sv} k(h_l(\boldsymbol{x_0}), sv) - \rho, \tag{1}$$

where α_{sv} is the coefficient of the support vector sv in the decision function, ρ is the intercept of the decision function and k is a Gaussian RBF kernel with kernel width γ:

$$k(\boldsymbol{x}, \boldsymbol{y}) = \exp\left(-\gamma \|\boldsymbol{x} - \boldsymbol{y}\|^2\right). \tag{2}$$

Hyperparameter optimization. The layer-specific scoring function takes two parameters as input: the regularization factor $\nu \in (0, 1)$ that represents an upper bound on the fraction of training errors (controlling for overfitting), and the kernel width γ. This hyperparameters must be carefully chosen to achieve good performances. For this purpose, many approaches have been proposed for hyperparameters selection in OCSVMs [57]. In our setting, we used the validation set of labeled examples $\mathcal{L}^{valid}$ to choose the best combination of parameters, based on the validation accuracy. To avoid a full (and infeasible) exploration of the parameters space, we employed Bayesian hyperparameter optimization, via the `scikit-optimize` library [56]. In Fig. 2, we report the estimated accuracy of the explored solutions in the specific case of the OCSVM detector, in a representative experimental setting.

[1] In a test on the DenseNet, CIFAR-10, CW scenario, the AUROC returned by the OCSVM detector with PCA-whitening preprocessing improves from 82.56 to 90.24, and the AUPR from 78.17 to 82.98, with respect to the same method without preprocessing (see Section 3 for further details).

Adversarial detection. Once the scores have been obtained for each layer, they can serve either as input for the standalone Detector A or as partial input of the ensemble detector ENAD. In the former case, in order to aggregate the scores of the separate layers $\mathbf{O}(\boldsymbol{x_0})$, this detector employs a logistic regression to model the posterior probability of adversarial (adv) examples:

$$p(adv|\mathbf{O}(\boldsymbol{x_0})) = \left(1 + \exp\left(\beta_0 + \boldsymbol{\beta}^T \mathbf{O}(\boldsymbol{x_0})\right)\right)^{-1}, \tag{3}$$

The parameters $\{\beta_0, \boldsymbol{\beta}\}$ are fitted with a cross-validated procedure using the labeled training set $\mathcal{L}^{train}$.

2.4. Detector B: Mahalanobis

The Mahalanobis detector (Maha) was originally introduced in [1]. The algorithmic procedure is akin to that of the OCSVM detector, and includes a final layer score aggregation step via logistic regression, but it is based on a different layer-specific scoring function.

Layer-specific scoring function. Given a test instance $\boldsymbol{x_0}$, the layer score is computed via a three-step procedure: first, for each instance, the class $\hat{c}$ is selected, such that:

$$\hat{c} = \arg\min_c \mathsf{MahaScore}_l(\boldsymbol{x_0}, c),$$

where $\mathsf{Maha}_l(\boldsymbol{x}, c)$ is the Mahalanobis distance for the lth layer between the activations $h_l(\boldsymbol{x})$ and the mean values $\mu_{l,c}$ of the examples in the training set $\mathcal{X}^{train}$:

$$\mathsf{MahaScore}_l(\boldsymbol{x}, c) = (h_l(\boldsymbol{x}) - \mu_{l,c})^T \boldsymbol{\Sigma}_l^{-1} (h_l(\boldsymbol{x}) - \mu_{l,c}), \tag{4}$$

where $\boldsymbol{\Sigma}_l$ is the covariance matrix of the examples of $\mathcal{X}^{train}$ in layer l. Then, the instance is preprocessed to obtain a better separation between benign and adversarial examples similar to what is discussed in [48]:

$$\boldsymbol{x}_0^* = \boldsymbol{x_0} - \lambda\, \mathrm{sign}\, \boldsymbol{\nabla}_{\boldsymbol{x_0}} \mathsf{MahaScore}_l(\boldsymbol{x_0}, \hat{c}),$$

where λ is a positive real number, called the perturbation magnitude. The scoring for instance $\boldsymbol{x_0}$ is computed as:

$$\mathbf{M}(\boldsymbol{x_0}) = [\mathrm{M}_1(\boldsymbol{x_0}), \mathrm{M}_2(\boldsymbol{x_0}), \ldots, \mathrm{M}_L(\boldsymbol{x_0})],$$

where

$$\mathrm{M}_l(\boldsymbol{x_0}) = -\max_c\ \mathsf{MahaScore}_l(\boldsymbol{x}_0^*, c).$$

Adversarial detection. The scores can serve either as input for the standalone Detector B or as partial input for the ensemble detector ENAD. In the latter case, the Mahalanobis detector uses logistic regression to identify adversarial examples, with a procedure similar to that already described for standalone Detector A.

Hyperparameter optimization. Differently from Detector A, the hyperparameter selection is performed downstream of the adversarial detection stage. In order to select the best λ (unique for all layers), the method selects the value that achieves the best Area Under the Receiver Operating Characteristic (AUROC, detailed in Section 2.7) on $\mathcal{L}^{valid}$ computed on the posterior probability $p(adv|\mathbf{M}(\boldsymbol{x_0}))$, which is obtained via the logistic regression fitted on $\mathcal{L}^{train}$.

2.5. Detector C: LID

The third detector uses a procedure similar to Detectors A-B, but the layer-specific scoring function is based on the Local Intrinsic Dimensionality (LID) approach [2].

Algorithm 2 ENAD detector.

```
Input: Act. $h_l$ of layer $l$, trainset $\mathcal{X}^{train}$, labeled set $\mathcal{L}$
1: Select best hyperparameters for OCSVM, Maha, LID
2: for each layer $l$ in $1, \ldots, L$ do
3:     Layer $l$ scores of $\boldsymbol{x_0}$: $\mathrm{O}_l(\boldsymbol{x_0}), \mathrm{M}_l(\boldsymbol{x_0}), \mathrm{L}_l(\boldsymbol{x_0})$
4: end for
5: Scores vector: $\mathbf{E}(\boldsymbol{x_0}) := [\mathbf{O}(\boldsymbol{x_0}), \mathbf{M}(\boldsymbol{x_0}), \mathbf{L}(\boldsymbol{x_0})]$
6: Fit $adv$ posterior on $\mathcal{L}^{train}$: $p(adv|\mathbf{E}(\boldsymbol{x_0}))$
7: ENAD on $\boldsymbol{x_0}$: $\mathsf{ENAD}(\boldsymbol{x_0}) := p(adv|\mathbf{E}(\boldsymbol{x_0}))$
8: return ENAD
```

Layer-specific scoring function. Given a test instance $\boldsymbol{x_0}$, the LID layer-specific scoring function L is defined as:

$$\mathrm{L}_l(\boldsymbol{x_0}) = -\left(\frac{1}{k}\sum_{i=1}^{k} \log \frac{r_i(h_l(\boldsymbol{x_0}))}{\max_i r_i(h_l(\boldsymbol{x_0}))}\right)^{-1}, \tag{5}$$

where, k is the number of nearest neighbors, r_i is the Euclidean distance to the ith nearest neighbor in the set of normal examples $\mathcal{L}_{norm}$. The layer-specific scores are:

$$\mathbf{L}(\boldsymbol{x_0}) = [\mathrm{L}_1(\boldsymbol{x_0}), \mathrm{L}_2(\boldsymbol{x_0}), \ldots, \mathrm{L}_L(\boldsymbol{x_0})]$$

Adversarial detection. When considered alone, the LID detector employs a logistic regression to identify adversarial examples, similarly to the other detectors (see above).

Hyperparameter optimization. Similarly to Detector B, the hyperparameter selection is performed downstream of the adversarial detection stage. k is selected as the value that achieves the best AUROC on $\mathcal{L}^{valid}$ computed on the posterior probability $p(adv|\mathbf{L}(\boldsymbol{x_0}))$, which is obtained via the logistic regression fitted on $\mathcal{L}^{train}$. Note that k is unique for all layers.

2.6. ENsemble adversarial detector (ENAD)

The ENAD approach exploits the effectiveness of Detectors A, B, and C in capturing different properties of data distributions, by explicitly integrating the distinct layer-specific scoring functions in a unique classification framework. More in detail, given a test instance $\boldsymbol{x_0}$, it will be characterized by a set of layer-specific and detector-specific features, computed from the scoring functions defined above, that is: $\mathbf{E}(\boldsymbol{x_0}) = [\mathbf{O}(\boldsymbol{x_0}), \mathbf{M}(\boldsymbol{x_0}), \mathbf{L}(\boldsymbol{x_0})]$. It should be noted that training and hyperparameter optimization is executed for each detector independently.

Adversarial detection. In its current implementation, in order to integrate the scores of the separate layers $\mathbf{E}(\boldsymbol{x_0})$, ENAD employs a simple logistic regression to model the posterior probability of adversarial (adv) examples:

$$p(adv|\mathbf{E}(\boldsymbol{x_0})) = \left(1 + \exp\left(\beta_0 + \boldsymbol{\beta}^T \mathbf{E}(\boldsymbol{x_0})\right)\right)^{-1}. \tag{6}$$

Like Detectors A, B, and C, the logistic is fitted with a cross-validation procedure using the labeled training set $\mathcal{L}^{train}$. Fitting the logistic allows one to have different weights, i.e. the elements of $\boldsymbol{\beta}^T$, for the different layers and detectors, meaning that a given detector might be more effective in isolating an adversarial example when processing its activation on a certain layer of the network. The pseudocode is reported in Algo. 2.

2.7. Performance metrics

Let the positive class be the adversarial examples (adv) and the negative class be the benign examples ($\overline{adv}$). Then, the correctly classified adversarial and benign examples correspond to the true positives (TP) and true negatives (TN), respectively. Conversely, the wrongly classified adversarial and benign examples are the false negatives (FN) and false positives (FP), respectively.

To evaluate the detectors performances, we employed two standard threshold independent metrics, namely AUROC and AUPR [58], the Accuracy and the F1-score, defined as follows:

- Area Under the Receiver Operating Characteristic curve (AUROC): the area under the curve identified by $\text{specificity} = \text{TN}/(\text{TN}+\text{FP})$ and $\text{fall}-\text{out} = \text{FP}/(\text{TN}+\text{FP})$.
- Area Under the Precision–Recall curve (AUPR): the area under the curve identified by Precision (Pr) $= \text{TP}/(\text{TP}+\text{FP})$ and Recall (Re) $= \text{TP}/(\text{TP}+\text{FN})$.
- Accuracy$= (\text{TP}+\text{TN})/(\text{TP}+\text{TN}+\text{FP}+\text{FN})$.
- F1-score (F1) $= 2 \times (\text{Pr} \times \text{Re})/(\text{Pr}+\text{Re})$.

The AUROC and AUPR were evaluated given the adversarial posterior learned by the logistic function $p(adv|X)$, where X is the set of layer-specific scores. The layer-specific scores are computed from $\mathcal{L}^{valid}$ when the AUROC is used for hyperparameters optimization for the Mahalanobis and LID detectors, and from $\mathcal{L}^{test}$ in all the other settings, i.e. the detector's performance evaluation. Moreover, AUROC and AUPR were also used to evaluate the performance of each detector in each layer, by considering the layer-specific scores evaluated on $\mathcal{L}^{test}$ (see, e.g., Fig. 3(b)). Note that for the OCSVM and Mahalanobis detectors, lower values correspond to adversarial examples, while the opposite applies to the LID detector. The accuracy was used for the Bayesian hyperparameter selection procedure [54] of the OCSVM detector. Lastly, F1-score, Precision and Recall where used to evaluate ensembling methods in Sect 3.1.3 and transfer attacks in Section 3.2.2.

3. Results

Four benchmark adversarial attacks were selected to test the effectiveness of our ENAD approach and competing methods, namely: FGSM [6], BIM [24], DeepFool [25] and CW [26]. In particular, to evaluate the performances in distinct scenarios, we designed two separate arrays of experiments:

1. *Known attacks* (Section 3.1): the same adversarial attack is employed both in the training and in the test phase, as proposed in [1].
2. *Transfer attacks* (transfer to *unknown attacks*, Section 3.2): a given attack is employed for training and another one for the test phase. In this case, two sub-scenarios were defined:
   a. *cheap training*, i.e., training on the FGSM attack and testing on the other three benchmark attacks.
   b. *hard attacks*, i.e., testing against high-confidence adversarial examples with many distortion levels generated using CW_∞.

We compared the performance of ENAD with standalone Detectors A (OCSVM), B (Mahalanobis [1]), and C (LID [2]). All four detectors integrate layer-specific anomaly scores via logistic regression and classify any example as adversarial if the posterior probability is > 0.5, benign otherwise (see the Methods for additional details). Moreover, we also evaluated the performance of the ExAD detector [13] and JTLA detector [33] in Section 3.1.4, and alternative ensembling strategies in the Known Attack scenario in Section 3.1.3.

In Section 3.3 we finally propose a strategy to visualize benign and adversarial examples on a low-dimensional space, based on the similarity of their layer- and detector-specific score profiles, and which may be useful in real-world applications. The assessment of the computational time of ENAD is discussed in Section 3.4, while the hyperparameter selection is discussed in Supplementary Sect. 1. In particular, Supplementary Table 1 contains the hyperparameter configurations and Supplementary Tables 4 and 5 contain the best hyperparameters for each setting.

3.1. Known attacks

3.1.1. Standalone detectors capture distinct properties of adversarial examples

In order to assess the ability of standalone Detectors A, B and C to exploit different properties of input instances, we first analyzed the methods as standalone, and computed the subsets of adversarial examples identified: (i) by all detectors, (ii) by a subset of them, (iii) by none of them.

In Fig. 3(a), we reported a contingency table in which we compare the OCSVM and Mahalanobis detectors on all the experimental settings with the DenseNet model, while the remaining pairwise comparisons are presented in the Supplementary Fig 4. Importantly, while the class of examples identified by both approaches is, as expected, the most crowded, we observe a substantial number of instances that are identified by either one of the two approaches. This important result appears to be general, as it is confirmed in the other comparisons between standalone detectors.

In addition, in Supplementary Fig. 5 one can find the layer-specific scores returned by all detectors in a specific setting (ResNet, DeepFool, CIFAR-10). For a significant portion of examples, the ranking ordering among scores is not consistent across detectors, confirming the distinct effectiveness in capturing different data properties in the hidden layers.

To investigate the importance of the layers with respect to the distinct attacks, models and datasets, we also computed the AUROC directly on the layer-specific scores, i.e. the anomaly scores returned by each detector in each layer. In Fig. 3(b), one can find the results for all detectors in all settings, with the DenseNet model (the same results for the ResNet model are available in Supplementary Fig. 3). For the FGSM attack, the scores computed on the middle layers consistently return the best AUROC in all datasets, while for the BIM attack the last layer is apparently the most important. Notably, with DeepFool and CW attacks the most important layers are dataset-specific. This result demonstrates that each attack may be vulnerable in distinct layers of the network.

3.1.2. ENAD outperforms standalone detectors

Table 1 reports the AUROC and AUPR computed on the fitted adversarial posterior probability for Detectors A, B, and C, respectively, on all 24 experimental settings.

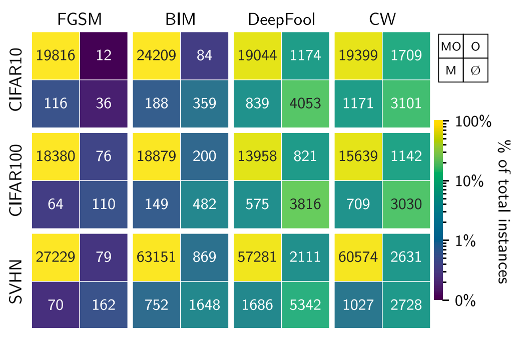
It can be noticed that ENAD exhibits the best AUROC, AUPR and F1-score in 21 out of 24 settings (with less than 1% difference from the best in the remaining ones), with the greatest improvements emerging in the hardest attacks, i.e. DeepFool and CW. Remarkably, the newly designed OCSVM detector outperforms the other standalone detectors in 12 out of 24 settings.

Notice that in Supplementary Table 6, we also evaluated the performance of all pairwise combinations of the three detectors, so to quantitatively investigate the impact of integrating the different algorithmic approaches, proving that distinct ensembles of detectors can be effective in specific experimental settings.

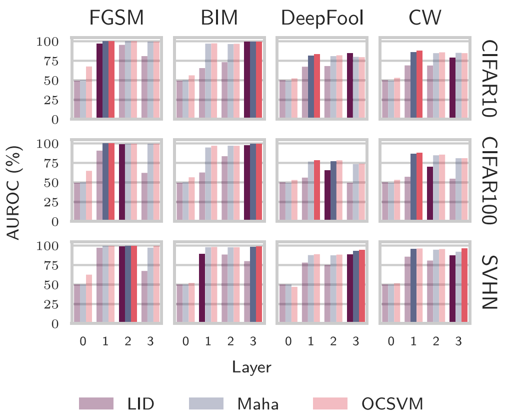
3.1.3. Comparison with alternative aggregation approaches

ENAD employs a logistic classifier as a meta-learner to aggregate the scoring of every detector in each layer, although other ensembling strategies may be employed, e.g., *voting schemes* or more complex machine learning algorithms. In this section, we will compare ENAD with (1) an AutoML framework named AutoGluon[2] [59], to get a strong baseline from a more complex machine learning algorithm than a simple logistic and (2) a simpler voting scheme. Both strategies will be employed to aggregate the predictions of standalone Detectors A, B, C.

[2] https://auto.gluon.ai/stable/index.html



(a)



(b)

Fig. 3. (a) Comparison of predictions of standalone detectors in the Known Attacks scenario (OCSVM vs. Mahalanobis – DenseNet). The contingency table shows the number of adversarial examples of the test set $\mathcal{L}^{test}$ correctly identified by: both the OCSVM and the Mahalanobis detectors (MO), either one of the two methods (O or M), none of them (∅). The results of the DenseNet model, with respect to the distinct datasets and attacks are shown, whereas the remaining pairwise comparisons are displayed in the Supplementary Fig. 4. **(b) Influence of the hidden layers in adversarial detection in the Known Attacks scenario (DenseNet).** For each configuration of datasets and attacks on the DenseNet model, the AUROC of each layer-specific score for Detectors A–C is returned. For each configuration and detector, the best-performing layer is highlighted with a darker shade (see Methods for further details). The same results for the ResNet model are available in Supplementary Fig. 3.

Table 1
Comparative assessment of ENAD and competing methods in the Known Attacks scenario. Performance comparison of the ENAD, LID [2], Mahalanobis [1], OCSVM detectors (all the pairwise combinations of the three single detectors are available in Supplementary Table 6). The Table contains the AUROC, AUPR and F1-score for all the combinations of selected datasets (CIFAR-10, CIFAR-100 and SVHN), models (DenseNet and ResNet), and attacks (FGSM, BIM, DeepFool and CW). See Methods for further details.

		Attack	FGSM			BIM			DeepFool			CW		
		Metric	AUPR	AUROC	F1-score	AUPR	AUROC	F1-score	AUPR	AUROC	F1-score	AUPR	AUROC	F1-score
Model	*Dataset*	*Detector*												
DenseNet	CIFAR-10	LID	96.37	98.30	91.00	99.52	99.73	96.76	75.21	85.22	65.04	69.75	80.88	57.72
		Maha	99.64	99.90	99.31	99.46	99.75	96.81	74.41	82.69	62.45	78.44	87.43	68.01
		OCSVM	99.59	99.90	98.55	99.18	99.67	96.55	77.02	84.74	68.59	82.99	90.25	75.26
		ENAD	**99.85**	**99.95**	**99.50**	**99.95**	**99.98**	**98.81**	**83.70**	**89.34**	**74.63**	**85.13**	**91.54**	**77.20**
	CIFAR-100	LID	98.39	99.29	94.21	96.31	98.11	90.00	55.88	70.12	40.07	59.66	72.80	44.16
		Maha	99.59	99.89	99.09	97.57	99.08	94.78	67.30	78.18	52.54	75.71	87.54	63.42
		OCSVM	99.49	99.84	98.62	98.23	99.30	95.28	69.17	79.27	59.49	78.70	88.95	70.89
		ENAD	**99.78**	**99.92**	**99.26**	**98.79**	**99.56**	**96.53**	**73.09**	**83.10**	**65.54**	**83.18**	**92.12**	**77.95**
	SVHN	LID	98.59	99.07	94.23	92.13	94.79	83.01	85.80	91.83	78.64	90.47	94.61	83.16
		Maha	99.46	99.85	98.74	97.94	99.26	94.27	89.93	94.93	81.97	91.03	97.17	87.72
		OCSVM	99.51	99.86	98.75	97.41	99.18	94.79	91.29	94.94	82.97	96.88	98.61	91.89
		ENAD	**99.84**	**99.94**	**99.10**	**99.04**	**99.59**	**95.85**	**93.30**	**96.02**	**86.23**	**98.28**	**99.22**	**94.10**
ResNet	CIFAR-10	LID	99.18	99.67	96.53	94.37	96.50	86.28	79.40	88.58	69.88	73.95	82.29	62.37
		Maha	99.90	99.93	99.59	99.06	99.58	96.14	85.64	91.60	75.19	92.28	95.90	84.55
		OCSVM	**99.99**	**99.99**	**99.76**	98.95	99.44	94.96	83.90	90.83	73.72	92.26	95.68	84.11
		ENAD	99.96	99.97	99.34	**99.59**	**99.79**	**97.26**	**88.17**	**92.91**	**78.40**	**93.26**	**96.38**	**85.63**
	CIFAR-100	LID	97.53	98.78	91.36	94.52	96.76	87.19	56.10	69.87	30.40	65.53	78.51	53.78
		Maha	99.49	99.72	**99.02**	93.51	96.92	87.25	73.32	85.23	63.54	83.00	91.68	74.92
		OCSVM	**99.63**	**99.86**	98.55	91.70	95.79	84.19	71.69	84.16	62.63	83.17	91.24	77.41
		ENAD	99.44	99.66	98.12	**98.40**	**99.28**	**95.21**	**76.78**	**86.34**	**67.10**	**88.31**	**94.09**	**81.28**
	SVHN	LID	94.52	97.84	89.07	83.46	90.78	73.90	86.60	92.31	76.49	79.46	88.16	70.56
		Maha	97.90	99.60	97.82	92.22	97.16	86.91	93.04	95.74	84.77	84.92	92.13	76.27
		OCSVM	98.05	99.65	**97.84**	95.93	98.13	90.03	92.16	95.59	84.96	89.19	93.30	79.81
		ENAD	**98.16**	**99.69**	97.67	**96.86**	**98.59**	**91.40**	**94.15**	**96.44**	**86.31**	**90.66**	**94.63**	**82.99**

Table 2
Comparative assessment of ENAD, AutoML and the Maj voting strategies in the Known Attacks scenario. The F1-score returned by ENAD, AutoML and the Maj(ority) voting scheme are shown for the Known Attacks scenario (see the main text for further details). The comparison with also the Or and And voting schemes is available in Supp. Tab. 8.

		Attack	FGSM	BIM	DeepFool	CW
		Metric	F1-score	F1-score	F1-score	F1-score
Model	*Dataset*	*Ensemble*				
DenseNet	CIFAR-10	ENAD	**99.50**	98.81	**74.63**	**77.20**
		AutoML	99.32	**98.90**	73.94	76.89
		Maj	98.88	97.26	67.15	71.76
	CIFAR-100	ENAD	99.26	**96.53**	**65.54**	77.95
		AutoML	**99.27**	96.49	65.28	**79.22**
		Maj	99.10	95.28	54.52	64.41
	SVHN	ENAD	99.10	95.85	86.23	94.10
		AutoML	**99.49**	**97.42**	**87.36**	**94.71**
		Maj	98.98	94.81	83.87	91.55
ResNet	CIFAR-10	ENAD	99.34	97.26	78.40	85.63
		AutoML	99.72	**97.47**	**78.77**	**86.30**
		Maj	**99.79**	96.30	74.93	84.23
	CIFAR-100	ENAD	98.12	95.21	67.10	81.28
		AutoML	98.80	**95.59**	**70.14**	**83.16**
		Maj	**98.87**	88.50	61.46	76.05
	SVHN	ENAD	97.67	91.40	86.31	82.99
		AutoML	97.89	**92.75**	**87.15**	**83.79**
		Maj	**97.92**	89.43	85.09	79.27

In detail, AutoGluon is an automated ML framework that employs well-known techniques, such as ensembling multiple classifiers (like we did in ENAD) and bagging to reduce overfitting, while employing heuristics for hyperparameter tuning and feature engineering. The classifiers taken into account include neural networks, boosted trees and k-nearest neighbors. Lastly, we ran AutoGluon with the present for best performance ("best_quality" preset to the `fit` method of the `TabularPredictor` class), keeping the default hyperparameter space and optimizing the AUROC metric. By doing so, we obtained an upper-bound estimate of the achievable performance using the layer-specific scores with more powerful machine learning algorithms.

Furthermore, we also considered simpler voting schemes as a lower-bound of performances: Or, And and Maj (Majority), that is, an example is identified as adversarial if *at least one* detector, *all* detectors or *the majority* of the detectors, respectively, classify it as adversarial.

In Table 2 we report the F1-score comparison among ENAD, AutoML and Maj in the Known attack setting (see above), while in Supplementary Table 8 we report all the voting schemes. In almost all the settings, ENAD and AutoML achieve the best performances. We point out that AutoML never outperforms ENAD and that voting scheme requires the training of three separate logistics, one for each detector, while ENAD requires only one. These results suggest that keeping the logistic aggregator is a good tradeoff between resource requirements and performance.

3.1.4. Comparison with similar adversarial detection approaches

In addition to LID and Mahalanobis, we also compared ENAD with two more adversarial detectors that are based on similar ideas to our method. First, JTLA [33] is an unsupervised adversarial detector that, akin to our approach, utilizes layer-specific scoring functions to calculate the anomaly score. Second, ExAD [13] is another ensemble-based detector that, in contrast to ENAD, relies on feature attribution methods [60–62] instead of the latent features.

Concerning JTLA, we tested its performance against ENAD with our experimental setting. We point out that our setup differs from theirs (described in [33]), since we have a smaller training set to fit detectors ($\mathcal{L}^{train}$) and we considered a different selection of layers in our DenseNet and ResNet models. The comparisons in the Known Attack scenario are available in Supplementary Table 2, while a more detailed description of the method and how we adapted our experimental setting can be found in Supplementary Sect. 2.1. To summarize, our method outperforms JTLA in all the combinations of the Known Attack scenario. We will later discuss that JTLA could have an advantage in transfer attacks, described in the next section, due to its unsupervised nature. Although, ENAD still performs better in the majority of the settings.

In the Supplementary Section 2.2, we provided a detailed description of ExAD and compared its performances to ENAD. As suggested by the authors of ExAD, either a CNN or an autoencoder (AE) can be employed to distinguish normal from adversarial feature attributions. In our experiments, we tested both CNNs and AEs for all benchmark attacks, the DenseNet model and the CIFAR-10 and SVHN datasets. Both types of classifiers achieved a maximum of ≈ 73 F1-score on the FGSM – DenseNet setting, while significantly worsened in all other cases (see Supplementary Table 3).

Table 3
Comparative assessment of ENAD and competing methods in the Transfer Attacks (cheap training) scenario. Performance comparison of the ENAD, LID [2], Mahalanobis [1], and OCSVM detectors when both the hyperparameter optimization and the logistic regression fit are performed on the FGSM attack. The Table contains the AUROC for all the combinations of selected datasets (CIFAR-10, CIFAR-100 and SVHN), models (DenseNet and ResNet), and attacks (BIM, DeepFool and CW). See Methods for further details.

		Detector	LID		Maha		OCSVM		ENAD	
Model	Dataset	Attack	AUPR	AUROC	AUPR	AUROC	AUPR	AUROC	AUPR	AUROC
DenseNet	CIFAR-10	BIM	88.28	92.97	98.00	99.24	98.93	99.59	**99.06**	**99.62**
		DeepFool	57.95	69.94	73.94	**82.88**	73.82	82.81	**74.72**	82.68
		CW	58.25	70.26	76.45	87.23	78.88	88.00	**79.96**	**88.40**
	CIFAR-100	BIM	28.62	31.45	**95.07**	**98.11**	90.94	95.84	57.70	60.27
		DeepFool	55.28	70.30	65.19	76.83	66.23	77.51	**68.90**	**79.21**
		CW	57.77	72.57	66.78	82.06	72.61	84.57	**73.86**	**84.71**
	SVHN	BIM	87.40	91.40	96.32	98.21	**97.26**	**99.14**	96.46	97.54
		DeepFool	73.50	79.50	86.76	91.17	**88.74**	**93.40**	85.59	88.58
		CW	75.48	84.63	88.43	94.33	**91.48**	**97.50**	88.62	92.16
ResNet	CIFAR-10	BIM	89.82	93.34	97.28	98.03	98.85	99.42	**99.02**	**99.47**
		DeepFool	61.62	73.92	75.55	81.34	79.23	**86.38**	**79.63**	85.66
		CW	67.79	78.05	79.01	84.75	**91.04**	**94.69**	90.01	94.55
	CIFAR-100	BIM	43.29	46.06	92.83	95.69	90.26	94.18	**93.70**	**96.68**
		DeepFool	55.23	68.12	68.31	78.24	72.08	**83.45**	**73.26**	83.26
		CW	64.89	76.24	81.72	88.70	80.53	87.56	**85.08**	**91.51**
	SVHN	BIM	71.38	85.03	88.10	95.15	**90.67**	**96.90**	84.54	93.38
		DeepFool	49.97	67.36	53.92	69.29	**59.65**	**75.11**	55.45	68.95
		CW	58.75	76.43	75.45	86.85	**80.03**	**89.70**	68.30	81.72

3.2. Transfer attacks

3.2.1. Transfer attacks with cheap training via FGSM

In this section, we discuss the performance of ENAD and Detectors A, B, and C when a transfer attack is performed. In particular, we tested the performance of all the detectors when trained on the cheapest benchmark attack, i.e. FGSM, that only requires the multiplication of the gradient by the perturbation size. Afterwards, we tested the performance on BIM, DeepFool and CW attacks.

In Table 3 one can see how, despite the expected worsening of the performances, either ENAD or OCSVM achieve the best AUROC and AUPR in almost all settings, further demonstrating the robustness of our approach.

When it comes to transfer attacks, it is also important to take into account the effectiveness of unsupervised techniques like JTLA, introduced in Section 3.1.4. Indeed, unlike ENAD, these methods do not depend on having a labeled training set in order to train the detectors. To estimate the performance of JTLA in the transfer setting, we can compare the results of Table 3 with the one in Supplementary Table 2 (the distinction between Transfer and Known attacks does not hold for JTLA). Our method still performs better than JTLA in 7 of 12 configurations, even if ENAD is in the cheap training setting.

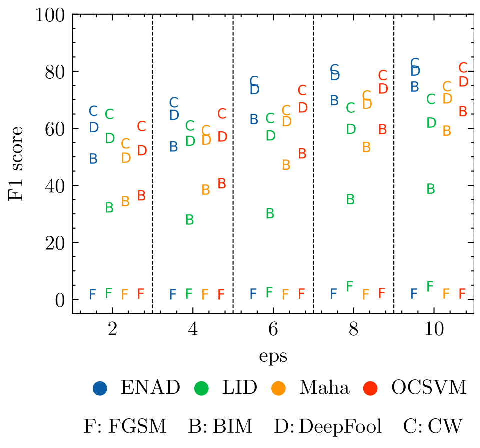
3.2.2. Training matters with harder transfer attacks

A particular kind of transfer attack is the so-called *adaptive attack*, where an attacker generates adversarial examples exploiting the full knowledge of the detector. Many works address the topic of adaptive attacks [26,28,63,64], in some cases by targeting ensembles of detectors [65]. In particular, in [28] the authors discuss how to design an effective attack when a defence method has some gradient masking. The LID detector falls in this category, as its loss function proves to be particularly difficult to differentiate.

Given that ENAD is an ensemble of multiple detectors, making its loss function is *at least* as difficult as that of the LID detector. We here applied the same strategy proposed in [28] to define a *harder* attack. In particular, we considered the L_∞ variant of the CW attack (labeled as CW_∞), which allows one to generate high-confidence adversarial examples with small distortions.

To this end, we generated 800 further adversarial examples with CW_∞ and distortion ϵ, and by scanning $\epsilon \in \{i/255 \mid i \in \{2,4,6,8,10\}\}$. This setting allowed us to assess the performance of the distinct detectors with respect to the distortion size. Also in this case, we defined three balanced partitions of adversarial, noisy, and clean examples (labels are coherent with the other experiments, see above), and tested ENAD and competing methods when trained on either FGSM, BIM, DeepFool, or CW, and tested against the CW_∞ attack.

In Fig. 4(a) we report the F1-score for the DenseNet and SVHN setting. All detectors perform better when trained either on DeepFool or CW, with ENAD showing the overall best performance for all values of ϵ. All detectors appear to be unable to classify CW_∞ adversarial examples when trained on FGSM (cheap training), which in this case



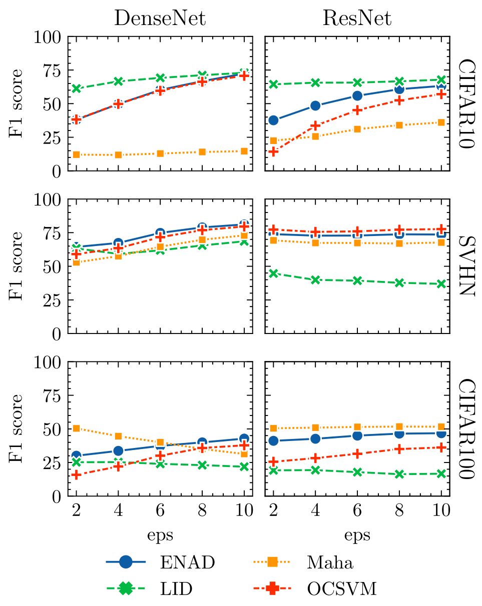


Fig. 4. Comparative assessment of ENAD and competing methods in the Transfer Attacks (hard attacks) scenario. (a) The of F1-score returned by ENAD, LID [2], Mahalanobis [1], and OCSVM detectors against the CW_∞ transfer attack is shown (DenseNet – SVHN). The dataset employed in this test contains 800 samples of, respectively, adversarial, noisy and clean examples (see the main text for further details). Colors are used to distinguish the detectors, and letters to distinguish the attack on which the detector is trained, e.g., the blue C stands for ENAD trained on CW. The remaining settings are shown in the Supplementary Fig. 6. **(b)** Analysis restricted on the performance of detectors when trained on the CW attack. In both figures, eps stands for the L_∞ attack max perturbation size (in the $[0, 255]$ scale).

is not advisable. All the other settings are reported in Supplementary Fig. 6.

In Fig. 4(b) we report the F1-score for all settings when the detectors are trained on CW, which appears to be the best choice as for Fig. 4(a). ENAD is the best performing in the DenseNet – SVHN setting, while being the second best in all remaining settings, showing good overall performance. OCSVM has analogous performances, while the other detectors exhibit unstable performances, confirming the results of Section 3.2.1.

3.3. Visualizing adversarial examples in the low-dimensional score space

In order to explore the relationship between the scoring functions and the overall performance of ENAD, it is possible to visualize the test examples on the low-dimensional tSNE space [66], using the (logit weighted and Z-scored) layer- and detector-specific scores as starting features (3 detectors $\times$ 4 hidden layers = 12 initial dimensions).

This representation allows one to intuitively assess how similar the score profiles of the test examples are: closer data points are those displaying more similar score profiles, which translates in an analogous distance from the set of correctly classified training instances, with respect to the three scoring functions currently included in ENAD. Importantly, this allows one to evaluate how many and which adversarial examples display score profiles closer than those of benign ones, and vice versa, and visualize them.

As an example, in Fig. 5 the test set of the SVHN, DenseNet, DeepFool setting is displayed on the tSNE space. The color gradient returns the confidence c of ENAD, i.e., the probability of the logistic regression: an example is categorized as adversarial if $c > 0.5$, benign otherwise.

While most of the adversarial examples are identified with high confidence (leftmost region of the tSNE plot), a narrow region exists in which adversarial examples overlap with benign ones, hampering their identification and leading to significant rates of both false positives and false negatives. Focusing on the set of false negatives, it is evident that some adversarial examples are scattered in the midst of the set of benign instances (rightmost region of the tSNE plot), rendering their identification extremely difficult.

3.4. Computational time

The computational time of ENAD is approximately the sum of that of the detectors employed to compute the layer-specific scores and, in the current version, it is mostly affected by OCSVM (see the computation time assessment in the Supplementary Material and in Supplementary Fig. 2). OCSVM hyperparameter search time can be improved in many ways: by considering SVM implementations that support GPU [67] or by adding a Hyperband scheduler [68], in addition to the Bayesian sampler that we employed in our experiments.

4. Conclusions

We introduced the ENAD ensemble approach for adversarial detection, motivated by the observation that distinct detectors are able to isolate non-overlapping subsets of adversarial examples, by exploiting different properties of the input data in the internal representation of a DNN. Accordingly, the integration of layer-specific scores extracted from three independent detectors (LID, Mahalanobis and OCSVM) allows ENAD to achieve significantly improved performance on benchmark datasets, models and attacks, with respect to the state-of-the-art, even when a simple integration scheme (i.e., logit) is adopted.

It is also worth of note that the newly introduced OCSVM detector proved highly effective as a standalone in our tests, indicating that the use of one-class classifiers for this specific task deserves an in-depth exploration. Most important, the theoretical framework of ENAD is designed to be general and as simple as possible, so to show the advantages of adopting ensemble approaches in the “cleanest” scenario. Yet, the framework might be easily extended and improved.

On the one hand, ENAD may accommodate different scoring functions, generated via any arbitrary set of independent algorithmic strategies. In this regard, ongoing efforts aim at integrating detectors processing the hidden layer features with others processing the properties of the *output*, which have already proven their effectiveness in adversarial detection (see, e.g. [14,42,48]). Similarly, one may explore the possibility of exploiting the information on activation paths and/or regions,

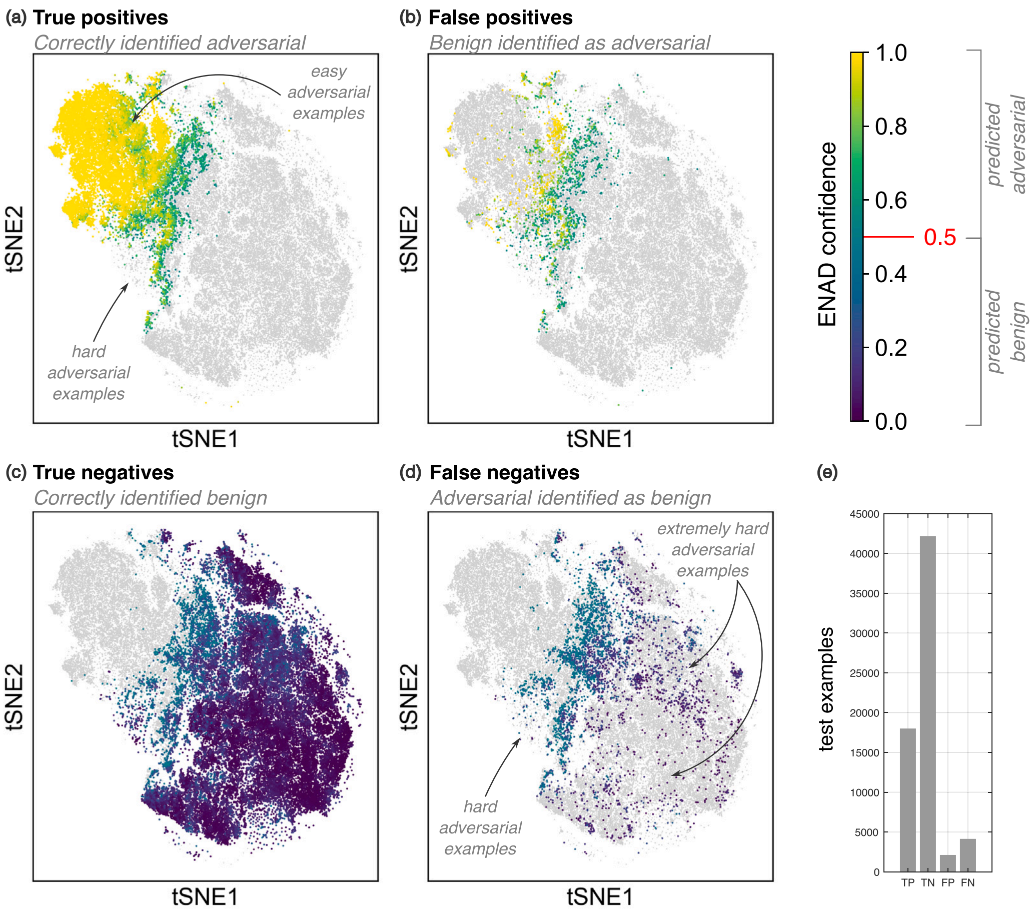


Fig. 5. Visualization of adversarial and benign examples in the low-dimensional score space (DenseNet – SVHN – DeepFool). Representation of the test data $\mathcal{L}^{test}$ for the configuration DeepFool, DenseNet and SVHN in the tSNE low-dimensional space [66]. The layer- and detector-specific scores are weighted with the logit weights, Z-scored, and then used as features for the tSNE computation, via the computation of the k-nearest neighbor graph ($k = 50$) with Pearson correlation as metric (further tSNE parameters: perplexity = 100, early exaggeration = 100, learning rate = 10000). In each quadrant the true positives **(a)**, false positives **(b)**, true negatives **(c)** and false negatives **(d)** are displayed. Each point in the plot represents either an adversarial or a benign example and the color returns the confidence c provided by ENAD, i.e., the probability returned by the logistic classifier. (e) The barplots return the absolute number of TPs, TNs, FPs and FNs for this experimental setting.

as suggested in [39,69], as well as of refining the score definition by focusing on class-conditioned features.

On the other hand, more effective strategies for the integration of such scoring functions might be devised. As shown the Results Section, adversarial examples generated with a given attack might be more easily identified by a specific detector and by exploiting the properties of a specific layer. In other terms, the attack type is closely related to the detector performance and the layer relevance, and this results in attack-specific optimal weights of the logit currently employed by ENAD and competing methods, possibly limiting its effectiveness. This aspect is even more relevant when facing transfer attacks, for which the logit training is executed on a separate attack, worsening the overall performance. This also suggests that the training phase should be considered with extreme caution when developing a detector for production.

For such reasons, more sophisticated strategies to combine scoring functions might be considered to improve the generality and robustness of our approach, e.g. via weighted averaging or by employing test statistics [33], as well as via the exploitation of more robust feature selection and classification strategies.

On a side note, we specify that, despite their simplicity, ensembling strategies based on voting schemes might be also considered as an alternative to the logit in safety-critical settings. For example, as presented in the Results Section, the Or voting scheme may be the method of choice if Recall matters more than Precision, as in several real-world biomedical scenarios (e.g., one might want to minimize the false negatives in diagnostic testing).

To conclude, given the virtually limitless possibility of algorithmic extensions of our framework, the superior performance exhibited in benchmark settings in a purposely simple implementation, and the theoretical and application expected impact, we advocate a widespread and timely adoption of ensemble approaches in the field of adversarial detection.

CRediT authorship contribution statement

Francesco Craighero: Conceptualization, Methodology, Software, Formal analysis, Data curation, Writing – original draft, Writing – review & editing, Visualization. **Fabrizio Angaroni:** Conceptualization, Methodology, Formal analysis. **Fabio Stella:** Conceptualization, Writing – review & editing. **Chiara Damiani:** Conceptualization, Methodology, Writing – review & editing. **Marco Antoniotti:** Conceptualization, Writing – review & editing, Funding acquisition. **Alex Graudenzi:** Conceptualization, Methodology, Writing – original draft, Writing – review & editing, Visualization, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

ENAD is freely available at: https://github.com/BIMIB-DISCo/ENAD-experiments, where one can also find the code to reproduce all the experiments presented in the article.

Acknowledgments

We thank Lucrezia Patruno, Davide Maspero, Bruno Giovanni Galuzzi, Gianluca Ascolani, Daniele Ramazzotti and Giancarlo Mauri for the useful discussions.

Fundings

This work has been partially supported by the CRUK/AIRC/FC-AECC Accelerator Award #22790 “Single Cell Cancer Evolution in the Clinic”, by the Horizon European Commission Program PPPA2027, PPPA-2021-AIPC #LC-01815952/101052609 “Towards an UNIque approach for artificial intelligence data-driven solutions to fight Childhood cAncer FOR Europe” (UNICA4EU), by the 2021 FAQC program of the Università degli Studi di Milano-Bicocca, by the MUR under the grant “Dipartimenti di Eccellenza 2023-2027” of the Department of Informatics, Systems and Communication of the University of Milano-Bicocca, Italy and by the Google Cloud Research Credits program.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.neucom.2023.126576.

References

[1] K. Lee, K. Lee, H. Lee, J. Shin, A simple unified framework for detecting out-of-distribution samples and adversarial attacks, in: S. Bengio, H.M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, R. Garnett (Eds.), Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada, 2018, pp. 7167–7177.

[2] X. Ma, B. Li, Y. Wang, S.M. Erfani, S.N.R. Wijewickrema, G. Schoenebeck, D. Song, M.E. Houle, J. Bailey, Characterizing adversarial subspaces using local intrinsic dimensionality, in: 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings, ICLR, Vancouver, BC, Canada, 2018.

[3] A. Krizhevsky, I. Sutskever, G.E. Hinton, ImageNet classification with deep convolutional neural networks, Commun. ACM 60 (6) (2017) 84–90, http://dx.doi.org/10.1145/3065386.

[4] A.W. Senior, R. Evans, J. Jumper, J. Kirkpatrick, L. Sifre, T. Green, C. Qin, A. Zídek, A.W.R. Nelson, A. Bridgland, H. Penedones, S. Petersen, K. Simonyan, S. Crossan, P. Kohli, D.T. Jones, D. Silver, K. Kavukcuoglu, D. Hassabis, Improved protein structure prediction using potentials from deep learning, Nature 577 (7792) (2020) 706–710, http://dx.doi.org/10.1038/s41586-019-1923-7.

[5] 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, ICLR, Banff, Canada, 2014.

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

[7] B. Kolosnjaji, A. Demontis, B. Biggio, D. Maiorca, G. Giacinto, C. Eckert, F. Roli, Adversarial malware binaries: Evading deep learning for malware detection in executables, in: 26th European Signal Processing Conference, EUSIPCO 2018, Roma, Italy, September 3-7, 2018, IEEE, Manhattan, New York, U.S., 2018, pp. 533–537, http://dx.doi.org/10.23919/EUSIPCO.2018.8553214.

[8] Y. Qin, N. Carlini, G.W. Cottrell, I.J. Goodfellow, C. Raffel, Imperceptible, robust, and targeted adversarial examples for automatic speech recognition, in: K. Chaudhuri, R. Salakhutdinov (Eds.), Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA, in: Proceedings of Machine Learning Research, vol. 97, PMLR, Long Beach, California, USA, 2019, pp. 5231–5240.

[9] N. Papernot, P.D. McDaniel, X. Wu, S. Jha, A. Swami, Distillation as a defense to adversarial perturbations against deep neural networks, in: IEEE Symposium on Security and Privacy, SP 2016, San Jose, CA, USA, May 22-26, 2016, IEEE Computer Society, Washington, D.C., United States, 2016, pp. 582–597, http://dx.doi.org/10.1109/SP.2016.41.

[10] M. Abbasi, C. Gagné, Robustness to adversarial examples through an ensemble of specialists, in: 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Workshop Track Proceedings, OpenReview.net, 2017.

[11] A. Bagnall, R. Bunescu, G. Stewart, Training ensembles to detect adversarial examples, 2017, arXiv:1712.04006.

[12] T. Strauss, M. Hanselmann, A. Junginger, H. Ulmer, Ensemble methods as a defense to adversarial perturbations against deep neural networks, 2018, arXiv:1709.03423.

[13] R. Vardhan, N. Liu, P. Chinprutthiwong, W. Fu, Z. Hu, X.B. Hu, G. Gu, ExAD: An ensemble approach for explanation-based adversarial detection, 2021, arXiv:2103.11526.

[14] K. Roth, Y. Kilcher, T. Hofmann, The odds are odd: A statistical test for detecting adversarial examples, in: K. Chaudhuri, R. Salakhutdinov (Eds.), Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA, in: Proceedings of Machine Learning Research, vol.97, PMLR, Long Beach, California, USA, 2019, pp. 5498–5507.

[15] T.G. Dietterich, Ensemble methods in machine learning, in: J. Kittler, F. Roli (Eds.), Multiple Classifier Systems, First International Workshop, MCS 2000, Cagliari, Italy, June 21-23, 2000, Proceedings, in: Lecture Notes in Computer Science, vol.1857, Springer, 2000, pp. 1–15, http://dx.doi.org/10.1007/3-540-45014-9_1.

[16] C.C. Aggarwal, S. Sathe, Outlier Ensembles - an Introduction, Springer, New York City, 2017, http://dx.doi.org/10.1007/978-3-319-54765-7.

[17] R. Kaur, S. Jha, A. Roy, S. Park, O. Sokolsky, I. Lee, Detecting OODs as datapoints with high uncertainty, 2021, CoRR abs/2108.06380, arXiv:2108.06380.

[18] B. Schölkopf, R.C. Williamson, A.J. Smola, J. Shawe-Taylor, J.C. Platt, Support vector method for novelty detection, in: S.A. Solla, T.K. Leen, K.-R. Müller (Eds.), Advances in Neural Information Processing Systems 12, NIPS Conference, Denver, Colorado, USA, November 29 - December 4, 1999, The MIT Press, Cambridge, Massachusetts, United States, 1999, pp. 582–588.

[19] S. Ma, Y. Liu, G. Tao, W.-C. Lee, X. Zhang, NIC: Detecting adversarial samples with neural network invariant checking, in: 26th Annual Network and Distributed System Security Symposium, NDSS 2019, San Diego, California, USA, February 24-27, 2019, The Internet Society, Reston, Virginia, United States, 2019.

[20] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, CVPR, 2016.

[21] G. Huang, Z. Liu, L. Van Der Maaten, K.Q. Weinberger, Densely connected convolutional networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 4700–4708.

[22] A. Krizhevsky, Learning multiple layers of features from tiny images, Univ. Toronto, 2009.

[23] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, A.Y. Ng, Reading digits in natural images with unsupervised feature learning, in: NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011, 2011.

[24] A. Kurakin, I.J. Goodfellow, S. Bengio, Adversarial examples in the physical world, in: 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Workshop Track Proceedings, ICLR, Toulon, France, 2017.

[25] S.-M. Moosavi-Dezfooli, A. Fawzi, P. Frossard, DeepFool: A simple and accurate method to fool deep neural networks, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016, IEEE Computer Society, Washington, D.C., United States, 2016, pp. 2574–2582, http://dx.doi.org/10.1109/CVPR.2016.282.

[26] 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.), Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security, AISec@CCS 2017, Dallas, TX, USA, November 3, 2017, ACM, Broadway, New York, 2017, pp. 3–14, http://dx.doi.org/10.1145/3128572.3140444.

[27] N. Carlini, D.A. Wagner, Towards evaluating the robustness of neural networks, in: 2017 IEEE Symposium on Security and Privacy, SP 2017, San Jose, CA, USA, May 22-26, 2017, IEEE Computer Society, Washington, D.C., United States, 2017, pp. 39–57, http://dx.doi.org/10.1109/SP.2017.49.

[28] A. Athalye, N. Carlini, D. Wagner, Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples, in: Proceedings of the 35th International Conference on Machine Learning, PMLR, 2018, pp. 274–283.

[29] D. Hendrycks, K. Gimpel, Early methods for detecting adversarial images, in: 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Workshop Track Proceedings, ICLR, Toulon, France, 2017.

[30] K. Grosse, P. Manoharan, N. Papernot, M. Backes, P.D. McDaniel, On the (statistical) detection of adversarial examples, 2017, CoRR abs/1702.06280, arXiv:1702.06280.

[31] F. Carrara, F. Falchi, R. Caldelli, G. Amato, R. Fumarola, R. Becarelli, Detecting adversarial example attacks to deep neural networks, in: Proceedings of the 15th International Workshop on Content-Based Multimedia Indexing, CBMI 2017, Florence, Italy, June 19-21, 2017, ACM, Broadway, New York, 2017, pp. 38:1–38:7, http://dx.doi.org/10.1145/3095713.3095753.

[32] N. Papernot, P. McDaniel, Deep K-nearest neighbors: Towards confident, interpretable and robust deep learning, 2018, arXiv:1803.04765.

[33] J. Raghuram, V. Chandrasekaran, S. Jha, S. Banerjee, A general framework for detecting anomalous inputs to DNN classifiers, in: M. Meila, T. Zhang (Eds.), Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event, in: Proceedings of Machine Learning Research, vol.139, PMLR, Virtual, 2021, pp. 8764–8775.

[34] G. Cohen, G. Sapiro, R. Giryes, Detecting adversarial samples using influence functions and nearest neighbors, in: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020, IEEE, Manhattan, New York, U.S., 2020, pp. 14441–14450, http://dx.doi.org/10.1109/CVPR42600.2020.01446.

[35] A. Abusnaina, Y. Wu, S. Arora, Y. Wang, F. Wang, H. Yang, D. Mohaisen, Adversarial example detection using latent neighborhood graph, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, ICCV, 2021, pp. 7687–7696.

[36] R. Kamoi, K. Kobayashi, Why is the mahalanobis distance effective for anomaly detection? 2020, CoRR abs/2003.00402, arXiv:2003.00402.

[37] Z. Zheng, P. Hong, Robust detection of adversarial attacks by modeling the intrinsic properties of deep neural networks, in: S. Bengio, H.M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, R. Garnett (Eds.), Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, MontrÉal, Canada, 2018, pp. 7924–7933.

[38] J.H. Metzen, T. Genewein, V. Fischer, B. Bischoff, On detecting adversarial perturbations, in: 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings, ICLR, Toulon, France, 2017.

[39] J. Lu, T. Issaranon, D.A. Forsyth, SafetyNet: Detecting and rejecting adversarial examples robustly, in: IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017, IEEE Computer Society, Washington, D.C., United States, 2017, pp. 446–454, http://dx.doi.org/10.1109/ICCV.2017.56.

[40] R. Feinman, R.R. Curtin, S. Shintre, A.B. Gardner, Detecting adversarial samples from artifacts, 2017, CoRR abs/1703.00410, arXiv:1703.00410.

[41] A. Sotgiu, A. Demontis, M. Melis, B. Biggio, G. Fumera, X. Feng, F. Roli, Deep neural rejection against adversarial examples, EURASIP J. Inf. Secur. 2020 (2020) 5, http://dx.doi.org/10.1186/s13635-020-00105-y.

[42] D. Hendrycks, K. Gimpel, A baseline for detecting misclassified and out-of-distribution examples in neural networks, in: 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings, ICLR, Toulon, France, 2017.

[43] W. Xu, D. Evans, Y. Qi, Feature squeezing: Detecting adversarial examples in deep neural networks, in: 25th Annual Network and Distributed System Security Symposium, NDSS 2018, San Diego, California, USA, February 18-21, 2018, The Internet Society, Reston, Virginia, United States, 2018.

[44] D. Meng, H. Chen, MagNet: A two-pronged defense against adversarial examples, in: B.M. Thuraisingham, D. Evans, T. Malkin, D. Xu (Eds.), Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017, ACM, Broadway, New York, 2017, pp. 135–147, http://dx.doi.org/10.1145/3133956.3134057.

[45] B. Liang, H. Li, M. Su, X. Li, W. Shi, X. Wang, Detecting adversarial image examples in deep neural networks with adaptive noise reduction, IEEE Trans. Dependable Secure Comput. 18 (1) (2021) 72–85, http://dx.doi.org/10.1109/TDSC.2018.2874243.

[46] B. Huang, Y. Wang, W. Wang, Model-agnostic adversarial detection by random perturbations, in: S. Kraus (Ed.), Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019, IJCAI, Macao, China, 2019, pp. 4689–4696, http://dx.doi.org/10.24963/ijcai.2019/651.

[47] F. Zuo, Q. Zeng, Exploiting the sensitivity of L2 adversarial examples to erase-and-restore, in: J. Cao, M.H. Au, Z. Lin, M. Yung (Eds.), ASIA CCS '21: ACM Asia Conference on Computer and Communications Security, Virtual Event, Hong Kong, June (2021) 7-11, ACM, 2021, pp. 40–51, http://dx.doi.org/10.1145/3433210.3437529.

[48] S. Liang, Y. Li, R. Srikant, Enhancing the reliability of out-of-distribution image detection in neural networks, in: 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings, ICLR, Vancouver, BC, Canada, 2018.

[49] J. Aigrain, M. Detyniecki, Detecting adversarial examples and other misclassifications in neural networks by introspection, 2019, CoRR abs/1905.09186, arXiv:1905.09186.

[50] J. Lust, A.P. Condurache, GraN: An efficient gradient-norm based detector for adversarial and misclassified examples, in: 28th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, ESANN 2020, Bruges, Belgium, October 2-4, 2020, 2020, pp. 7–12.

[51] J. Monteiro, I. Albuquerque, Z. Akhtar, T.H. Falk, Generalizable adversarial examples detection based on Bi-model decision mismatch, in: 2019 IEEE International Conference on Systems, Man and Cybernetics, SMC 2019, Bari, Italy, October 6-9, 2019, IEEE, Manhattan, New York, U.S., 2019, pp. 2839–2844, http://dx.doi.org/10.1109/SMC.2019.8913861.

[52] D. Tax, One-class classification; concept-learning in the absence of counter-examples (Ph.D. thesis), Delft University of Technology, 2001.

[53] A. Kessy, A. Lewin, K. Strimmer, Optimal whitening and decorrelation, Amer. Statist. 72 (4) (2018) 309–314, http://dx.doi.org/10.1080/00031305.2016.1277159.

[54] J. Snoek, H. Larochelle, R.P. Adams, Practical Bayesian optimization of machine learning algorithms, in: P.L. Bartlett, F.C.N. Pereira, C.J.C. Burges, L. Bottou, K.Q. Weinberger (Eds.), Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a Meeting Held December 3-6, 2012, Lake Tahoe, Nevada, United States, 2012, pp. 2960–2968.

[55] C.C. Aggarwal, Linear Algebra and Optimization for Machine Learning - a Textbook, Springer, New York City, 2020, http://dx.doi.org/10.1007/978-3-030-40344-7.

[56] T. Head, M. Kumar, H. Nahrstaedt, G. Louppe, I. Shcherbatyi, Scikit-Optimize: Sequential Model-Based Optimization in Python, Zenodo, http://dx.doi.org/10.5281/zenodo.1157319.

[57] S. Alam, S.K. Sonbhadra, S. Agarwal, P. Nagabhushan, One-class support vector classifiers: A survey, Knowl.-Based Syst. 196 (2020) 105754, http://dx.doi.org/10.1016/j.knosys.2020.105754.

[58] J. Davis, M. Goadrich, The relationship between precision-recall and ROC curves, in: W.W. Cohen, A.W. Moore (Eds.), Machine Learning, Proceedings of the Twenty-Third International Conference, ICML 2006, Pittsburgh, Pennsylvania, USA, June 25-29, 2006, in: ACM International Conference Proceeding Series, vol.148, ACM, Pittsburgh, Pennsylvania, USA, 2006, pp. 233–240, http://dx.doi.org/10.1145/1143844.1143874.

[59] N. Erickson, J. Mueller, A. Shirkov, H. Zhang, P. Larroy, M. Li, A.J. Smola, AutoGluon-tabular: Robust and accurate autoML for structured data, 2020, CoRR abs/2003.06505, arXiv:2003.06505.

[60] J.T. Springenberg, A. Dosovitskiy, T. Brox, M.A. Riedmiller, Striving for simplicity: The all convolutional net, in: Y. Bengio, Y. LeCun (Eds.), 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Workshop Track Proceedings, 2015.

[61] A. Shrikumar, P. Greenside, A. Kundaje, Learning important features through propagating activation differences, in: Proceedings of the 34th International Conference on Machine Learning, PMLR, 2017, pp. 3145–3153.

[62] M. Sundararajan, A. Taly, Q. Yan, Axiomatic attribution for deep networks, in: Proceedings of the 34th International Conference on Machine Learning, PMLR, 2017, pp. 3319–3328.

[63] W. He, J. Wei, X. Chen, N. Carlini, D. Song, Adversarial example defense: Ensembles of weak defenses are not strong, in: 11th USENIX Workshop on Offensive Technologies, WOOT 17, 2017.

[64] F. Tramèr, N. Carlini, W. Brendel, A. Madry, On adaptive attacks to adversarial example defenses, in: H. Larochelle, M. Ranzato, R. Hadsell, M.-F. Balcan, H.-T. Lin (Eds.), Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, Virtual, 2020.

[65] W. He, J. Wei, X. Chen, N. Carlini, D. Song, Adversarial example defense: Ensembles of weak defenses are not strong, in: 11th USENIX Workshop on Offensive Technologies, WOOT 17, 2017.

[66] L. van der Maaten, G. Hinton, Visualizing data using T-SNE, J. Mach. Learn. Res. 9 (86) (2008) 2579–2605.

[67] Z. Wen, J. Shi, Q. Li, B. He, J. Chen, ThunderSVM: A fast SVM library on GPUs and CPUs, J. Mach. Learn. Res. 19 (2018) 797–801.

[68] S. Falkner, A. Klein, F. Hutter, BOHB: robust and efficient hyperparameter optimization at scale, in: J.G. Dy, A. Krause (Eds.), Proceedings of the 35th International Conference on Machine Learning, ICML 2018, StockholmsmäSsan, Stockholm, Sweden, July 10-15, 2018, in: Proceedings of Machine Learning Research, vol.80, PMLR, 2018, pp. 1436–1445, URL http://proceedings.mlr.press/v80/falkner18a.html.

[69] F. Craighero, F. Angaroni, A. Graudenzi, F. Stella, M. Antoniotti, Investigating the compositional structure of deep neural networks, in: G. Nicosia, V. Ojha, E. La Malfa, G. Jansen, V. Sciacca, P. Pardalos, G. Giuffrida, R. Umeton (Eds.), Machine Learning, Optimization, and Data Science, Springer International Publishing, Cham, 2020, pp. 322–334.

Francesco Craighero is a Postdoctoral Researcher in the Signal Processing Laboratory 2 (LTS2) at the École polytechnique fédérale de Lausanne (EPFL) in Lausanne, Switzerland. He got his Master's degree in Computer Science at the University of Udine in 2018. In 2019, he joined the Data and Computational Biology lab (DCB) at the University of Milano-Bicocca, where he accomplished his Ph.D. in Computer Science in 2023. His main areas of interest are interpretability and anomaly detection in Deep Learning, but he has also worked on multidisciplinary projects in computational biology and cancer data science.

Fabrizio Angaroni is a Postdoc Research Fellow at the Center of Computational Biology of the Human Technopole in Milan. He previously worked as a Postdoc at the Department of Informatics, Systems, and Communication of the University of Milano-Bicocca. He holds a Ph.D. in Physics and Astrophysics from the University of Insubria. He is an author of over 20 scientific publications in indexed journals and conference proceedings. His current research is devoted to population genetics, stochastic and nonlinear dynamical systems, control theory applied to biological systems, statistical inference from omics data, and mathematical evolution, with an inclination to numerical optimization methods.

Fabio Stella is Associate Professor at the University of Milano-Bicocca, where he leads the model and algorithms for data and text mining (MADLAB) research lab. His main research interests are related to Bayesian networks and Probabilistic Graphical Models for finance, health, and biology. He published 100+ papers and served as Program Chair/Reviewer of several international conferences: ICLR, ICML, IJCAI, PGM, NeurIPS, PAKDD, RecSys, and UAI. He has been awarded the 10% best reviewers at NeurIPS2020 and 2022, AISTATS2022 and ICML2022. He is associate editor of IEEE Intelligent Systems and has been PI of several research projects.

Chiara Damiani is Associate Professor in Computer Science at the Department of Biotechnology and Biosciences of the University of Milan-Bicocca. She received her Ph.D. in computational modeling and simulation in 2011 from the University of Modena and Reggio Emilia. She is associate editor of BMC Bioinformatics. She is coordinator of the Community of Special Interest (COSI) in Computational Modelling of Biological Systems (SysMod) of the International Society of Systems Biology (ISCB). She is author of 50+ publications in Systems Biology and Bioinformatics, with a special focus on computational models of cancer metabolism.

Marco Antoniotti is Full Professor at the University of Milan-Bicocca, where he is Group Leader of the Data and Computational Biology (DCB) lab. Previously he worked at NYU Courant Bioinformatics Group, PARADES EEIG and University of California. He received his Ph.D. in Computer Science from the NYU. His research topics are Bioinformatics, Computational Biology and Artificial Intelligence recently applied to Cancer Research. He is author of several journal and conference papers and of software projects, and has received support from Regione Lombardia, NSF (USA), the Elixir European network, the European Commission (Horizon Europe, MCSA, COST) and the CRUK/AIRC Accelerator Program.

Alex Graudenzi is Assistant Professor at the Univ. of Milan-Bicocca. He is co-head of the Data and Computational Biology Lab and Director of the Como School on Cancer Evolution. He received his Ph.D. in computational modeling in 2010. Since then, he has been Research Associate, Assistant Professor and Visiting Scientist at several Research Centers and Universities. He is author of 65+ publications on journals and conference proceedings, recipient of research awards, collaborator in 10+ international projects, coauthor of 15+ tools for bioinformatics, and program committee member of several international conferences. His work combines Data Science, Artificial Intelligence and Complex Systems to investigate biological complexity and evolution.