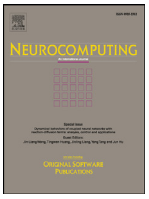
Contents lists available at ScienceDirect
Neurocomputing
journal homepage: www.elsevier.com/locate/neucom



Detecting adversarial manipulation using inductive Venn-ABERS predictors

Jonathan Peck[a,b,*], Bart Goossens[c], Yvan Saeys[a,b]

[a] Department of Applied Mathematics, Computer Science and Statistics, Ghent University, Ghent 9000, Belgium
[b] Data Mining and Modeling for Biomedicine, VIB Inflammation Research Center, Ghent 9052, Belgium
[c] Department of Telecommunications and Information Processing, Ghent University, Ghent 9000, Belgium

ARTICLE INFO

Article history:
Received 9 July 2019
Revised 29 October 2019
Accepted 4 November 2019
Available online xxx

Keywords:
Adversarial robustness
Conformal prediction
Supervised learning
Deep learning

ABSTRACT

Inductive Venn-ABERS predictors (IVAPs) are a type of probabilistic predictors with the theoretical guarantee that their predictions are perfectly calibrated. In this paper, we propose to exploit this calibration property for the detection of adversarial examples in binary classification tasks. By rejecting predictions if the uncertainty of the IVAP is too high, we obtain an algorithm that is both accurate on the original test set and resistant to adversarial examples. This robustness is observed on adversarials for the underlying model as well as adversarials that were generated by taking the IVAP into account. The method appears to offer competitive robustness compared to the state-of-the-art in adversarial defense yet it is computationally much more tractable.

1. Introduction

Modern machine learning methods are able to achieve exceptional empirical performance in many classification tasks [1,2]. However, they usually give only *point predictions*: a typical machine learning algorithm for classification of, say, images from the ImageNet data set [3] will output only a single label given any image without reporting how accurate this prediction is. In complex domains such as medicine, it is highly desirable to have not just a prediction but a measure of confidence in this prediction as well. Many state-of-the-art machine learning methods provide some semblance of such a measure by transforming their inputs into vectors of probabilities over the different possible outputs and then selecting as the point prediction the output with the highest probability. Usually, these probabilities are obtained by first training a *scoring classifier* which assigns to each input a vector of scores, one score per class. These scores are then turned into probabilities via some method of *calibration*. The most popular calibration method is Platt's scaling [4], which fits a logistic sigmoid to the scores of the classifiers.

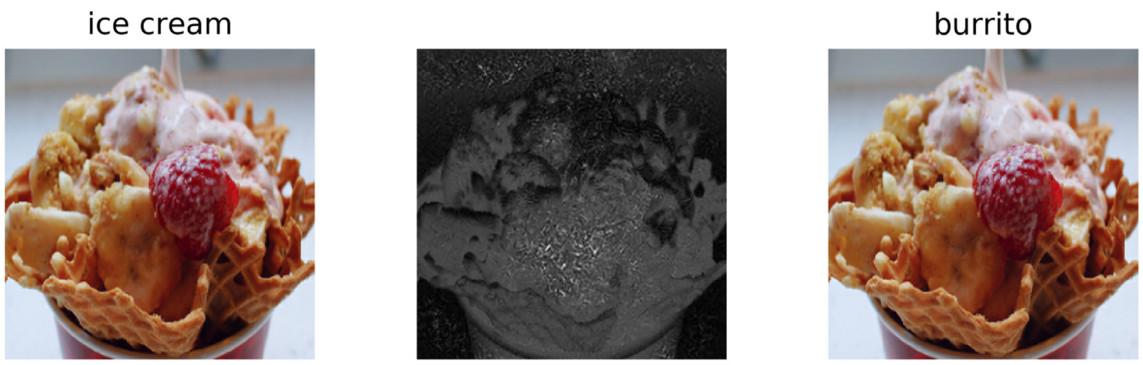
Contrary to popular belief, however, the probability vectors that result from methods such as Platt's scaling cannot be reliably interpreted as measures of confidence [5]. The phenomenon of *adversarial manipulation* illustrates this problem: for virtually all current machine learning classifiers, it is possible to construct *adversarial examples* [6]. These are inputs which clearly belong to a certain class according to human observers and which are highly similar to inputs on which the model performs very well. Despite this, the model misclassifies the adversarial example and assigns a very high probability to the resulting (incorrect) label. In fact, the inputs need not even be similar to any natural input; machine learning models will also sometimes classify unrecognizable inputs as belonging to certain classes with very high confidence [7]. Fig. 1 shows examples both of imperceptible adversarial perturbations as well as unrecognizable images which are classified as familiar objects with high confidence. These observations clearly undermine the idea that one can use such calibrated scores as measures of model confidence. It begs the question:

> Do there exist machine learning algorithms which yield provably valid measures of confidence in their predictions? If so, could these models be used to detect adversarial manipulation of input data?

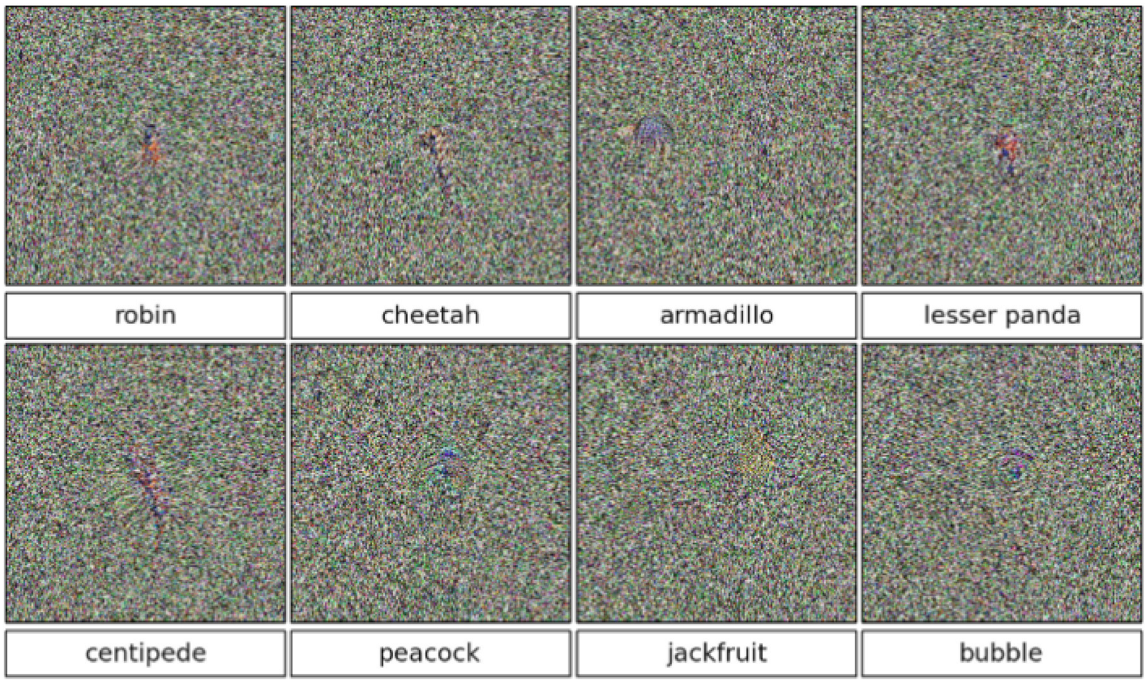
The answer to the first question is known to be affirmative: the algorithms in question are called *conformal predictors* [8–10]. Our contribution here is to show that such valid confidence measures can in fact be used to detect adversarial examples. In particular, there exist methods which can turn any scoring classifier into a probabilistic predictor with validity guarantees; this construction is

* Corresponding author at: Department of Applied Mathematics, Computer Science and Statistics, Ghent University, Ghent 9000, Belgium.
E-mail addresses: jonathan.peck@ugent.be (J. Peck), bart.goossens@ugent.be (B. Goossens), yvan.saeys@ugent.be (Y. Saeys).

https://doi.org/10.1016/j.neucom.2019.11.113
0925-2312/© 2020 The Author(s). Published by Elsevier B.V. This is an open access article under the CC BY license. (http://creativecommons.org/licenses/by/4.0/)



(a) Example of an imperceptible adversarial perturbation. The image on the left is correctly classified as `ice cream` by the ResNet50 deep neural network [12]. Applying the perturbation shown in the middle leads to the image on the right, which is visually indistinguishable from the original. However, the network now classifies it as `burrito`. The original image is part of the ImageNet data set [3]. The perturbation was generated using an unpublished method by the first author. It works by approximating the Jacobian of the network and adding a scaled version of its principal singular vector to the image.



(b) Examples of unrecognizable images classified as familiar objects by state-of-the-art neural networks with high confidence ($> 99\%$). These images were taken from Nguyen et al. [7].

Fig. 1. Illustrations of adversarial and fooling images. These examples highlight the fact that the confidence scores output by deep neural network classifiers are unreliable.

known as an *inductive Venn-ABERS predictor* or IVAP [11]. By making use of the confidence estimates output by the IVAPs, many state-of-the-art machine learning models can be made robust to adversarial manipulation.

1.1. Related work

The reliability of machine learning techniques in adversarial settings has been the subject of much research for a number of years already [12–15]. Early work in this field studied how a linear classifier for spam could be tricked by carefully crafted changes in the contents of spam e-mails, without significantly altering the readability of the messages. More recently, Szegedy et al. [14] showed that deep neural networks also suffer from this problem. Since this work, research interest in the phenomenon of adversarial examples has increased substantially and many attacks and defenses have been proposed [6,16,17]. The defenses can be broadly categorized as follows:

- Detector methods [18–21]. These defenses construct a *detector* which augments the underlying classification model with the capability to detect whether an input is adversarial or not. If the detector signals a possible adversarial, the model prediction is considered unreliable on that instance and the classification result is flagged.
- Denoising methods [22–25]. Here, the goal is to restore the adversarial examples to their original, uncorrupted versions and then perform classification on these cleaned samples.
- Other methods [6,22,26–28]. Another class of defenses performs neither explicit detection nor filtering. Rather, the aim is to make the model inherently more robust to manipulation via

data augmentation, regularization, modified optimization algorithms or special architectures.

Despite the large number of defenses that have been proposed so far, at the time of this writing only one technique is generally accepted as having any noticable effect [29]: adversarial training [27,28]. However, even this method currently has too limited success. The Madry defense [27], for instance, achieves less than 50% adversarial accuracy on the CIFAR-10 data set [30] even though state-of-the-art clean accuracy is over 95% [31]. Moreover, many recently proposed defenses suffer from a phenomenon called *gradient masking* [32]. Here, the defense protects the model against adversarials by obfuscating its gradient information. This is commonly done by introducing non-differentiable components such as randomization or JPEG compression. However, such tactics only render the model robust against gradient-based attacks which crucially rely on this information to succeed. Attacks that do not need gradients or that can cope with imprecise approximations (such as BPDA [32] or black-box attacks [33,34]) will not be deterred by gradient masking defenses.

We are not the first to consider the hypothesis that adversarial examples are due to faulty confidence measures. Most existing research in this area has focused on utilizing the uncertainty estimates provided by Bayesian deep learning [35], especially Monte Carlo dropout [36–38] and other stochastic regularization techniques [5]. On the other hand, our approach is decidedly frequentist and hence differs significantly from this related work. We are not aware of much other work in this area that is also frequentist (e.g. Grosse et al. [20]), as the field of deep learning (and model uncertainty in particular) appears to be primarily Bayesian. Although significant progress is being made [39], scaling the Bayesian methods to state-of-the-art deep neural networks remains an open problem. The Bayesian approach of integrating over weighted likelihoods also suffers from certain pathologies which may diminish its usefulness in the adversarial setting [40]. It is our hope that the method detailed here, which draws upon the powerful framework of conformal prediction [9], can serve as a scalable and effective alternative to Bayesian deep learning.

The defense we propose in this work falls under the category of detector methods. We are well-aware of the bad track record that these methods have: for example, Carlini and Wagner [41] evaluate many recently proposed adversarial detector methods and find that they can be easily bypassed. However, by following the advice of Carlini and Wagner [41] and Athalye et al. [32] in the evaluation of our detector, we hope to show convincingly that our method has the potential to be a strong, scalable and relatively simple defense against adversarial manipulation. In particular, we test our detector on adversarial examples generated using existing attacks that take the detector into account as well as a novel white-box attack that is specifically tailored to our defense. We conclude that the defense we propose in this work remains robust even when the attacks are adapted to the defense and it appears to be competitive with the defense proposed by Madry et al. [27].

The present work is an extension of our ESANN 2019 submission [42]. The main additions are as follows:

- Include the Zeroes vs ones data set.
- Perform an ablation study to test the sensitivity of the IVAP to its hyperparameters.
- Evaluate the defense in the ℓ_2 norm as well instead of only ℓ_∞.
- The discussions of the results has been significantly extended.
- We have released a reference implementation on GitHub[1].
- Include qualitative examples of adversarials produced by our white-box attack.

[1] https://github.com/saeyslab/binary-ivap

1.2. Organization

The rest of the paper is organised as follows. Section 2 provides the necessary background about supervised learning, conformal prediction and IVAPs which will be used in the sequel. Section 3 describes the defense mechanism which we propose for the detection of adversarial inputs. This defense is experimentally evaluated on several tasks in Section 4. Experimental comparisons to the Madry defense are carried out in Section 5. Section 6 contains the conclusion and avenues for future work. The code for our reference implementation can be found at

https://github.com/saeyslab/binary-ivap

2. Background

We consider the typical supervised learning setup for classification. There is a measurable *object space* $\mathcal{X}$ and a measurable *label space* $\mathcal{Y}$. We let $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$. There is an unknown probability measure P on $\mathcal{Z}$ which we aim to estimate. In particular, we have a class of $\mathcal{X} \to \mathcal{Y}$ functions $\mathcal{H}$ and a data set $S = \{(x_i, y_i) \mid i = 1, \dots, m\}$ of i.i.d. samples from P. Our goal is to find a function in $\mathcal{H}$ which fits the data best:

$$f^\star = \operatorname{argmin}_{f \in \mathcal{H}} R_S(f).$$

Here, R_S is the *empirical risk*

$$R_S(f) = \frac{1}{m} \sum_{i=1}^{m} \ell(x_i, y_i, f),$$

where ℓ is the *loss function*. In principle, this can be any non-negative $\mathcal{X} \times \mathcal{Y} \times \mathcal{H} \to \mathbb{R}$ function. Its purpose is to measure how "close" the function f is to the ground-truth y_i locally at the point x_i. Our objective is to find a minimizer $f^\star$ in $\mathcal{H}$ for the average of the loss over the data set S. For k-ary classification, a commonly used loss function is the *log loss*

$$\ell_{\log}(x, y, f) = -\log p_y(x; f).$$

Here, $p_y(x; f)$ is the probability that f assigns to class y for the input x. Ideally, for a discriminative model this should be equal to the conditional probability $\Pr[y \mid x]$ under the measure P. In the case of a *scoring classifier*, these probabilities are computed by fitting a scoring function g to the data followed by a calibration procedure s. The final classification is then computed as

$$f(x) = \operatorname{argmax}_{i=1,\dots,k} s_i(g(x)). \quad (1)$$

In the case of Platt's scaling [4], which is in fact a logistic regression on the classifier scores, s_i is the softmax function

$$s_i(z) = \frac{\exp(w_i^\intercal z + b_i)}{\sum_j \exp(w_j^\intercal z + b_j)}.$$

Here, w_i and b_i are learned parameters.

2.1. Conformal prediction

It is known [5,36] that the calibrated scores produced by scoring classifiers such as (1) cannot actually be used as reliable estimates of the conditional probability $\Pr[y \mid x]$: it is possible to generate inputs for any given classifier such that the model consistently assigns high probability to the wrong classes. It makes sense, then, to look for a type of machine learning algorithm which has provable guarantees on the confidence of its predictions. This leads us naturally to the study of *conformal predictors* [8], which hold exactly this promise.

In general, a conformal predictor is a function Γ which, when given a confidence level $\varepsilon \in [0, 1]$, a bag[2] of instances $B =$

[2] A *bag* or *multiset* is a collection of objects where the order is unimportant (like a set) and duplicates are allowed (like a list).

$\wr z_1, \ldots, z_n \wr$ and a new object $x \in \mathcal{X}$, outputs a set $\Gamma^\varepsilon(B, x) \subseteq \mathcal{Y}$. Intuitively, this set contains those labels which the predictor believes with at least $1-\varepsilon$ confidence could be the true label of the input sample based on the bag of examples given to it. These predictors must satisfy two properties [8,10]:

1. Nestedness. If $\varepsilon_1 \geq \varepsilon_2$ then $\Gamma^{\varepsilon_1}(B, x) \subseteq \Gamma^{\varepsilon_2}(B, x)$.
2. Exact validity. If the true label of x is y, then $y \in \Gamma^\varepsilon(B, x)$ with probability at least $1-\varepsilon$.

The exact validity property is too much to ask from a deterministic predictor [8]. Instead, the algorithms we consider here are only *conservatively valid*. Specifically, let $\omega = z_1, z_2, \ldots$ be an infinite sequence of samples from an *exchangeable* distribution P. A distribution is said to be exchangeable if for every sequence $z_1, \ldots, z_n$ and permutation ϕ of the integers $\{1, \ldots, n\}$ it holds that

$$\Pr[z_1, \ldots, z_n] = \Pr[z_{\phi(1)}, \ldots, z_{\phi(n)}].$$

That is, each permutation of a sequence of samples from P is equally likely. In particular, i.i.d. samples are always exchangeable. Define the error after seeing n samples at significance level ε as

$$\mathrm{err}_n^\varepsilon(\Gamma, \omega) = \begin{cases} 1 & if y_n \notin \Gamma^\varepsilon(\wr z_1, \ldots, z_{n-1} \wr, x_n), \\ 0 & \text{otherwise}. \end{cases}$$

The quantities $\mathrm{err}_n^\varepsilon(\Gamma, \omega)$ are random variables depending on $\omega \sim P^\infty$. Exact validity means these variables are all independent and Bernoulli distributed with parameter ε. Conservative validity means they are *dominated in distribution* by independent Bernoulli random variables. That is, there exist two sequences of random variables $\xi_1, \xi_2, \ldots$ and $\eta_1, \eta_2, \ldots$ such that

1. each ξ_n is independent and Bernoulli distributed with parameter ε;
2. $\mathrm{err}_n^\varepsilon(\Gamma, \omega) \leq \eta_n$ almost surely for all n;
3. the joint distribution of $\eta_1, \ldots, \eta_n$ equals that of $\xi_1, \ldots, \xi_n$ for all n.

This implies the following asymptotic conservative validity property, by the law of large numbers:

$$\lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^n \mathrm{err}_i^\varepsilon(\Gamma, \omega) \leq \varepsilon \text{ almost surely.}$$

Algorithm 1 is the general conformal prediction algorithm. It

Algorithm 1: The conformal prediction algorithm.

```
Input: Non-conformity measure $\Delta$, significance level $\varepsilon$, bag
       of examples $\wr z_1, \ldots, z_n \wr \subseteq \mathcal{Z}$, object $x \in \mathcal{X}$
Output: Prediction region $\Gamma^\varepsilon(\wr z_1, \ldots, z_n \wr, x)$
1 foreach $y \in \mathcal{Y}$ do
2     $z_{n+1} \leftarrow (x, y)$
3     for $i = 1, \ldots, n+1$ do
4         $\alpha_i \leftarrow \Delta(\wr z_1, \ldots, z_n \wr \setminus \wr z_i \wr, z_i)$
5     end
6     $p_y \leftarrow \frac{1}{n+1}\#\{i = 1, \ldots, n+1 \mid \alpha_i \geq \alpha_{n+1}\}$
7 end
8 return $\{y \in \mathcal{Y} \mid p_y > \varepsilon\}$
```

takes as a parameter a *non-conformity measure* Δ. In general, a non-conformity measure is any measurable real-valued function which takes a sequence of samples $z_1, \ldots, z_n$ along with an additional sample z and maps them to a *non-conformity score* $\Delta(\wr z_1, \ldots, z_n \wr, z)$. This score is intended to measure how much the sample z differs from the given bag of samples. The conformal prediction algorithm is always conservatively valid regardless of the choice of non-conformity measure[3]. However, the *predictive efficiency* of the algorithm — that is, the size of the prediction region $\Gamma^\varepsilon(B, x)$ — can vary considerably with different choices for Δ. If the non-conformity measure is chosen sufficiently poorly, the prediction regions may even be equal to the entirety of $\mathcal{Y}$. Although this is clearly valid, it is useless from a practical point of view.

Algorithm 1 determines a prediction region for a new input $x \in \mathcal{X}$ based on a bag of old samples by iterating over every label $y \in \mathcal{Y}$ and computing an associated p-value p_y. This value is the empirical fraction of samples in the bag (including the new "virtual sample" (x, y)) with a non-conformity score that is at least as large as the non-conformity score of (x, y). By thresholding these p-values we obtain a set of candidate labels $y_1, \ldots, y_t$ such that each possible combination $(x, y_1), \ldots, (x, y_t)$ is "sufficiently conformal" to the old samples at the given level of confidence.

2.2. Inductive Venn-ABERS predictors

Of particular interest to us here will be the *inductive Venn-ABERS predictors* or IVAPs [11]. These are related to conformal predictors but they take advantage of the predictive efficiency of some other inductive learning rule (such as a neural network or support vector machine). The IVAP algorithm is shown in Algorithm 2 for

Algorithm 2: The inductive Venn-ABERS prediction algorithm for binary classification.

```
Input: bag of examples $\wr z_1, \ldots, z_n \wr \subseteq \mathcal{Z}$, object $x \in \mathcal{X}$,
       learning algorithm $A$
Output: Pair of probabilities $(p_0, p_1)$
1  Divide the bag of training examples $\wr z_1, \ldots, z_n \wr$ into a proper
   training set $\wr z_1, \ldots, z_m \wr$ and a calibration set $\wr z_{m+1}, \ldots, z_n \wr$.
2  Run the learning algorithm $A$ on the proper training set to
   obtain a scoring rule $F$.
3  foreach example $z_i = (x_i, y_i)$ in the calibration set do
4      $s_i \leftarrow F(x_i)$
5  end
6  $s \leftarrow F(x)$
7  Fit isotonic regression to $\{(s_{m+1}, y_{m+1}), \ldots, (s_n, y_n), (s, 0)\}$
   obtaining a function $f_0$.
8  Fit isotonic regression to $\{(s_{m+1}, y_{m+1}), \ldots, (s_n, y_n), (s, 1)\}$
   obtaining a function $f_1$.
9  $(p_0, p_1) \leftarrow (f_0(s), f_1(s))$
10 return $(p_0, p_1)$
```

the case of binary classification. The output of the IVAP algorithm is a pair (p_0, p_1) where $0 \leq p_0 \leq p_1 \leq 1$. These quantities can be interpreted as lower and upper bounds on the probability $\Pr[y = 1 \mid x]$, that is, $p_0 \leq \Pr[y = 1 \mid x] \leq p_1$. The width of this interval, $p_1 - p_0$, can be used as a reliable measure of confidence in the prediction. Although Algorithm 2 can only be used for binary classification, it is possible to extend it to the multi-class setting [44]. This is left to future work.

IVAPs are a variant of the conformal prediction algorithm where the non-conformity measure is based on an isotonic regression of the scores which the underlying scoring classifier assigns to the calibration data points as well as the new input to be classified. Isotonic (or monotonic) regression aims to fit a non-decreasing free-form line to a sequence of observations such that the line lies as close to these observations as possible. Fig. 2 shows an example of isotonic regression applied to a 2D toy data set.

[3] One might even use Bayesian methods in the computation of Δ. In particular, Δ might be computed using integration over weighted likelihoods determined by a parametric model together with some prior. This combination of Bayesian inference with frequentist methods has been called *Frasian inference* [43].

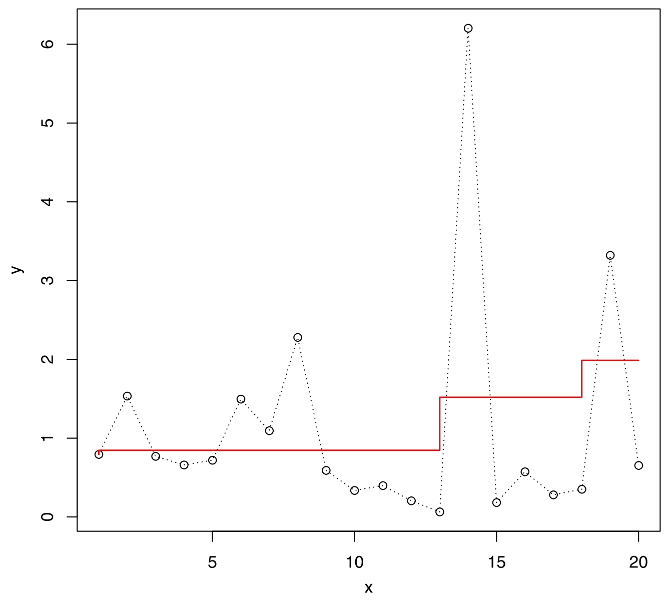


Fig. 2. An example of isotonic regression on a 2D data set. The isotonic fit is shown as a red line. This image was produced with the `isoreg` function from the R language [45].

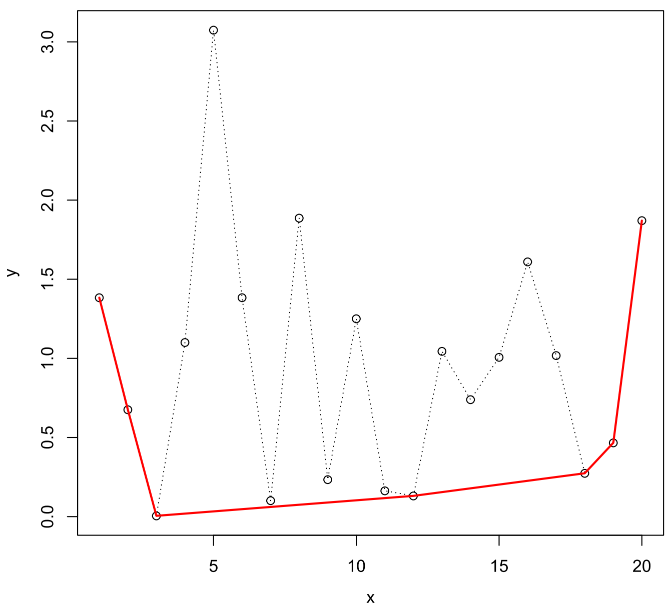


Fig. 3. Example of the greatest convex minorant of a set of points, shown as a red line. This image was produced with the `fdrtool` package [47].

In the case of Algorithm 2, the isotonic regression is performed as follows. Let $s_1, \dots, s_k$ be the scores assigned to the calibration points. First, these points are sorted in increasing order and duplicates are removed, obtaining a sequence $s'_1 \le \dots \le s'_t$. We then define the *multiplicity* of s'_j as

$$w_j = \#\{i \mid s_i = s'_j\}.$$

The "average label" corresponding to some score s'_j is

$$y'_j = \frac{1}{w_j} \sum_{i:s_i = s'_j} y_i.$$

The *cumulative sum diagram* (CSD) is computed as the set of points

$$P_i = \left(\sum_{j=1}^{i} w_j, \sum_{j=1}^{i} y'_j w_j \right) = (W_i, Y_i)$$

for $i = 1, \dots, t$. For these points, the *greatest convex minorant* (GCM) is computed. Fig. 3 shows an example of a GCM computed for a given set of points in the plane. Formally, the GCM of a function $f : U \to \mathbb{R}$ is the maximal convex function $g : I \to \mathbb{R}$ defined on a closed interval I containing U such that $g(u) \le f(u)$ for all $u \in U$ [46]. It can be thought of as the "lowest part" of the convex hull of the graph of f.

The value at s'_i of the isotonic regression is now defined as the slope of the GCM between W_{i-1} and W_i. That is, if f is the isotonic regression and g is the GCM, then

$$f(s'_i) = \frac{g(W_i) - g(W_{i-1})}{W_i - W_{i-1}} = \frac{g(W_i) - g(W_{i-1})}{w_i}.$$

We leave the values of f at other points besides the s'_i undefined, as we will never need them here.

3. Detecting adversarial manipulation using IVAPs

IVAPs enjoy a type of validity property which we detail here. A random variable P is said to be *perfectly calibrated* for another random variable Y if the following equality holds almost surely:

$$\mathbb{E}[Y \mid P] = P. \tag{2}$$

Let P_0, P_1 be random variables representing the output of an IVAP trained on a set of i.i.d. samples and evaluated on a new random sample X with label $Y \in \{0, 1\}$. Then there exists a random variable $S \in \{0, 1\}$ such that P_S is perfectly calibrated for Y. Hence, for each prediction at least one of the probabilites P_0, P_1 output by an IVAP is almost surely equal to the conditional expectation of the label given the probability. Since the label is binary, the relation (2) can be rewritten as

$$\Pr[Y = 1 \mid P_S] = P_S. \tag{3}$$

Eq. (3) expresses the validity property for IVAPs. If the difference $p_1 - p_0$ is sufficiently small for some instance x, so that $p_0, p_1 \approx p$, then (3) allows us to deduce that x has label 1 with probability approximately p. The validity property (3) holds for IVAPs as long as the data are exchangeable, which is the case whenever the data are i.i.d.

Our proposed method for detecting adversarial examples is based on these observations: we use $p_1 - p_0$ as a confidence measure for the prediction returned by the underlying scoring classifier and then return the appropriate label based on these values. The pseudocode is shown in Algorithm 3. The general idea is to use an IVAP to obtain the probabilities p_0 and p_1 for each test instance x. If these probabilities lie close enough to each other according to the precision parameter β, then we assume to have enough confidence in our prediction; otherwise, we return a special label `REJECT` signifying that we do not trust the output of the classifier on this particular sample. The precision parameter β is tuned by maximizing Youden's index [48] on a held-out validation set consisting of clean and adversarial samples. In case we trust our prediction, we use $p = \frac{p_1}{1-p_0+p_1}$ as an estimate of the probability that the label is 1 (as in Vovk et al. [11]). If $p > 0.5$, we return 1; otherwise, we return 0. This method of quantifying uncertainty and rejecting unreliable predictions was in fact already mentioned in Vovk et al. [11], but to our knowledge nobody has yet attempted to use it for protection against adversarial examples.

An important consideration in the deployment of Algorithm 3 is the computational complexity of the approach. This is clearly determined by the underlying machine learning model, as we are required to run its learning algorithm and perform inference with the resulting model for each sample in the calibration set as well as for each new test sample. The overhead caused by the IVAP

```
Algorithm 3: Detecting adversarial manipulation with IVAPs.
Input: precision $\beta \in [0, 1]$, bag of examples $\wr z_1, \ldots, z_n \wr \subseteq \mathcal{Z}$,
       object $x \in \mathcal{X}$, learning algorithm $A$
Output: An element of the set $\mathcal{Y} \cup \{\texttt{REJECT}\}$.
1  Use algorithm 2 to compute $p_0, p_1$ for $x$.
2  if $p_1 - p_0 \leq \beta$ then
3      Set $p \leftarrow \frac{p_1}{1-p_0+p_1}$.
4      if $p > 0.5$ then
5          return 1
6      else
7          return 0
8      end
9  else
10     return REJECT
11 end
```

construction consists of the isotonic regressions, which are dominated by the complexity of sorting the samples in the calibration set, as well as other operations that take linear time. The scores of the calibration samples can be precomputed and stored for fast access later, reducing the complexity of the inference step. To summarize, let $T_A(n)$ and $T_F(n)$ denote the time complexity of running the learning algorithm and the scoring rule on a set of n samples, respectively. Then we can determine the time complexity of Algorithm 3 as follows for a bag of n samples:

- *Calibration*. Initially, when the IVAP is calibrated, we run the learning algorithm on the proper training set and compute scores for each of the calibration samples. This takes $\mathcal{O}(T_A(n) + T_F(n))$ time.
- *Inference*. To compute the probabilities p_0, p_1 for a new test sample, we run the scoring rule and perform two isotonic regressions. This can be done in time $\mathcal{O}(T_F(1) + n \log n)$.

The overhead incurred by the IVAP in the calibration phase (which only needs to be performed once) is proportional to the complexity of the learning algorithm and scoring rule when executed on the given bag of samples. When performing inference, the overhead compared to simply running the underlying model is log-linear in the size of the calibration set.

It is also important to consider how the calibration set and the underlying model affect the performance of the IVAP. To the best of our knowledge, the existing literature on IVAPs does not provide quantitative answers to these questions. Vovk et al. [11] note that the validity property (3) always holds for the IVAP regardless of the performance of the underlying model, as long as the calibration set consists of independent samples from the data distribution. They also point out that, for "large" calibration sets, the difference $p_1 - p_0$ will be small. It would be interesting to have uniform convergence bounds that quantify how quickly $p_1 - p_0$ converges to zero as the calibration set grows larger, but we are not aware of any such results. Most likely, such bounds will depend on the accuracy of the underlying model, where the convergence is faster with a more accurate model.

3.1. White-box attack for IVAPs

As pointed out by Carlini and Wagner [41], it is not sufficient to demonstrate that a new defense is robust against existing attacks. To make any serious claims of adversarial robustness, we must also develop and test a white-box attack that was designed to specifically target models protected with our defense. Let x be any input that is not rejected and classified as y by the detector. Our attack must then find $\tilde{x} \in \mathcal{X}$ such that

1. $\|x - \tilde{x}\|_p$ is as small as possible;
2. $\tilde{x}$ is not rejected by the detector;
3. $\tilde{x}$ is classified as $1 - y$.

Here, $\|u\|_p$ is the general ℓ_p norm of the vector u. Common choices for p in the literature are $p \in \{1, 2, \infty\}$ [27]; we will focus on ℓ_2 and ℓ_∞ norms in this work.

Following Carlini and Wagner [41], we design a differentiable function that is minimized when $\tilde{x}$ lies close to x, is not rejected by the detector and is classified as $1 - y$. To do this efficiently, note that whenever a new sample has the same score as an old sample in the calibration set, it will have no effect on the isotonic regression and the result of applying Algorithm 3 to it will be the same as applying the algorithm to the old sample. Hence, we search for a sample (s_i, y_i) in the calibration set such that

1. $y_i = 1 - y$;
2. $f_1(s_i) - f_0(s_i) \leq \beta$;
3. the confidence $p = \frac{f_1(s_i)}{1 - f_0(s_i) + f_1(s_i)}$ satisfies $p > 0.5$ if $y_i = 1$ and $p \leq 0.5$ otherwise.

From among all samples that satisfy these conditions, we choose the one which minimizes the following expression:

$$\|x - x_i\|_2^2 + c(s(x) - s_i)^2.$$

Here, $s(x)$ is the score assigned to x by the classifier and c is a constant which trades off the relative importance of reducing the size of the perturbation vs fooling the detector. We choose these samples in this way so as to "warm-start" our attack: we seek a sample that lies as close as possible to the original x in $\mathcal{X}$-space but also has a score s_i which lies as close as possible to $s(x)$. This way, we hope to minimize the amount of effort required to optimize our adversarial example.

Having fixed such a sample (s_i, y_i), we solve the following optimization problem:

$$\min_{\delta \in [-1,1]^d} \|\delta\|_2^2 + c(s(x+\delta) - s_i)^2 \text{ subject to } x + \delta \in [0, 1]^d. \quad (4)$$

The constant c can be determined via binary search, as in the Carlini & Wagner attack [16]. In our case, we are applying the attack to a standard neural network without any non-differentiable components, meaning the score function s is differentiable. The resulting problem (4) can therefore be solved using standard gradient descent methods since the function to be minimized is differentiable as well. In particular, we use the Adam optimizer [49] to minimize our objective function. The constraint $x + \delta \in [0, 1]^d$ can easily be enforced by clipping the values of $x + \delta$ back into the $[0, 1]^d$ hypercube after each iteration of gradient descent.

Note that our custom white-box attack will not suffer from any gradient masking introduced by our defense. Indeed, we only rely on gradients computed from the underlying machine learning model. As long as the unprotected classifier does not mask gradients (which it should not, since the unprotected classifiers will be vanilla neural networks trained in the standard way), this information will be useful.

As described, the above method of selecting a calibration sample and solving (4) only considers adversarials that are minimal in ℓ_2 distance. However, ℓ_∞-bounded perturbations are also of interest, so in our experiments we evaluate both the original ℓ_2 formulation as well as an ℓ_∞ variant where we replace $\|\cdot\|_2^2$ by $\|\cdot\|_\infty$.

4. Experiments

To verify the robustness of Algorithm 3 against adversarial examples, we perform experiments with several machine learning classification tasks. The following questions are of interest to us here:

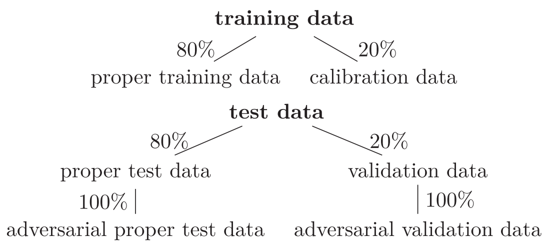


Fig. 4. Illustration of the different data splits. The training data is split into proper training data on which the scoring classifier is trained and calibration data on which the IVAP fits the isotonic regression. The test data is split into proper test data, which is used to determine the overall accuracy of the resulting algorithm, and validation data. Both of these subsets of test data are used to generate adversarial proper test data and adversarial validation data. The adversarial proper test data is used for evaluating the robustness of the resulting algorithm, whereas the adversarial validation data is used together with the validation data to tune β. All splits are 80% for the left branch and 20% for the right branch.

1. How does the IVAP handle adversarial examples targeted at the original model (Section 4.1)? Ideally, the IVAP should be immune to these.
2. Can we generate adversarial examples specifically to bypass the IVAP (Sections 4.2 and 4.3)? As pointed out by Carlini and Wagner [16], it is not enough for an adversarial defense to be robust against an oblivious adversary; we must also establish robustness against an adversary that is aware of our defense. We therefore adapt existing adversarial attacks to target our detector and we also evaluate the IVAP against our custom white-box attack.
3. How much adversarial data is actually needed in the validation set (Section 4.4)? As it stands, we augment the validation set with adversarial examples generated by a variety of attacks in order to tune the β parameter. However, we want our defense to be robust not only against attacks which it has been exposed to but also to attacks which have yet to be invented. We therefore compare our results to a setting where only one adversarial attack is used for the validation set.

The goal of these experiments is to show that the IVAP can handle both old and new adversarial examples and that its robustness generalizes beyond the limited set of attacks which it has seen. As such, we consider a number of different scenarios for each classification task, detailed below.

4.1. Robustness of the detector to the original adversarial examples

We run adversarial attack algorithms on both the validation set and the proper test set, creating an *adversarial validation set* and an *adversarial proper test set* (see Fig. 4 for details on our different data splits). The attacks we employed were projected gradient descent with random restarts [27], DeepFool [50], local search [51], the single pixel attack [52], NewtonFool [53], fast gradient sign [6] and the momentum iterative method [54]. This means for each test set we generate seven new data sets consisting of all the adversarial examples the attacks were able to find on the test set for the original model. We then run Algorithm 3 on the adversarial validation set as well as the regular validation set. Our goal is to maximize *Youden's index* (defined below) on both the regular and adversarial validation sets. We choose β such that this index is maximized. We can then use the proper test set and the adversarial proper test set to judge the clean and adversarial accuracy of the resulting algorithm with the tuned value of β.

4.2. Robustness of the detector to new adversarial examples

We attempt to generate new adversarials which take the detector into account and evaluate the performance similarly to the

Table 1
Summary of performance indicators for the CNNs used in our experiments.

Task	Clean accuracy	Adversarial accuracy
Zeroes vs ones	100%	2.31%
Cats vs dogs	89.87%	1.8%
T-shirts vs trousers	99.75%	3.43%
Airplanes vs automobiles	96.69%	3.75%

first scenario. The resulting samples are called *adapted adversarials*. They are generated by applying all of the adversarial attacks mentioned above to Algorithm 3. For the gradient-based attacks, we use the natural evolution strategies approach [34,55] to estimate the gradient, thereby avoiding any possible gradient masking issues [32].

4.3. Fully white-box attack

We run our custom adversarial attack (4) for the detector on the entire proper test set and evaluate its performance. This attack is specifically designed to target the IVAP and should represent an absolute worst-case scenario for the defense.

4.4. Ablation study

We estimate β based on adversarials generated only by the DeepFool method on the validation set for the original model and then evaluate the resulting detector as before. The purpose of this scenario is to estimate how "future-proof" our method is, by evaluating its robustness against attacks that played no role in the tuning of β. Our choice for DeepFool was motivated by the observations that it is efficient and rather effective, but it is by no means the strongest attack in our collection. It is less efficient than fast gradient sign, but this method was never meant to be a serious attack; it was only meant to demonstrate excessive linearity of DNNs [6]. It is weaker than NewtonFool, for example, since this method utilizes second-order information of the model whereas DeepFool is limited to first-order approximations. It is also weaker than some first-order attacks such as the momentum iterative method, which was at the top of the NIPS 2017 adversarial attack competition [54,56]. We therefore feel that DeepFool is a good choice for evaluating how robust our defense is to future attacks.

4.5. Metrics

For all scenarios, we report the accuracy, true prositive rate (TPR), true negative rate (TNR), false positive rate (FPR) and false negative rate (FNR) of the detectors. These are defined as follows:

$$\text{Accuracy} = \frac{\text{TA} + \text{TR}}{m}, \quad \text{TPR} = \frac{\text{TA}}{\text{TA} + \text{FR}}, \quad \text{TNR} = \frac{\text{TR}}{\text{TR} + \text{FA}},$$
$$\text{FPR} = \frac{\text{FA}}{\text{FA} + \text{TR}}, \quad \text{FNR} = \frac{\text{FR}}{\text{FR} + \text{TA}}.$$

Here, m is the total number of samples, TA is the number of correct predictions the detector accepted, TR is the number of incorrect predictions the detector rejected, FA is the number of incorrect predictions the detector accepted and FR is the number of correct predictions the detector rejected. Youden's index is defined as

$$J = \text{TPR} + \text{TNR} - 1 = \text{TPR} - \text{FPR}.$$

It is defined for every point on a ROC curve. Graphically, it corresponds to the distance between the ROC curve and the random chance line [48].

These metrics are computed for different scenarios:

- *Clean*. This refers to the proper test data set, without any modifications.

Table 2
Summary of performance indicators for the detectors on the different tasks.

Task	Data	Size	Accuracy	TPR	TNR	FPR	FNR
Zeroes vs ones	Clean	1692	98.52%	98.51%	100.0%	0.0%	1.49%
	Adversarial	6918	46.07%	0.0%	100.0%	0.0%	100.0%
	Adapted	5858	97.41%	0.0%	99.06%	0.94%	100.0%
	Custom ℓ_2	1692	0.0%	0.0%	0.0%	100.0%	100.0%
	Custom ℓ_∞	1692	0.0%	0.0%	0.0%	100.0%	100.0%
Cats vs dogs	Clean	3988	72.94%	75.9%	48.85%	51.15%	24.1%
	Adversarial	22,599	52.56%	71.25%	38.86%	61.14%	28.75%
	Adapted	17,187	59.27%	100.0%	59.18%	40.82%	0.0%
	Custom ℓ_2	3952	1.64%	100.0%	0.0%	100.0%	0.0%
	Custom ℓ_∞	3145	1.69%	100.0%	0.0%	100.0%	0.0%
T-shirts vs trousers	Clean	1600	96.88%	96.86%	97.44%	2.56%	3.14%
	Adversarial	7076	50.99%	0.0%	99.97%	0.03%	100.0%
	Adapted	5923	77.53%	0.0%	77.99%	22.01%	100.0%
	Custom ℓ_2	1600	0.0%	0.0%	0.0%	100.0%	100.0%
	Custom ℓ_∞	1600	0.0%	0.0%	0.0%	100.0%	100.0%
Airplanes vs automobiles	Clean	1600	74.38%	73.21%	96.3%	3.7%	26.79%
	Adversarial	8256	58.14%	0.0%	100.0%	0.0%	100.0%
	Adapted	6586	93.99%	0.0%	94.82%	5.18%	100.0%
	Custom ℓ_2	1600	0.0%	0.0%	0.0%	100.0%	100.0%
	Custom ℓ_∞	1600	0.0%	0.0%	0.0%	100.0%	100.0%

- *Adversarial*. Here, the metrics are computed on the adversarial test set. This data set is constructed by running existing adversarial attacks against the underlying neural network on the proper test set. They do not take the IVAP into account.
- *Adapted*. This is similar to the Adversarial scenario but the attacks are modified to take the IVAP into account.
- *Custom ℓ_p*. Here, we compute the metrics for adversarial examples generated using our custom ℓ_p white-box attack on the proper test set. We report results for $p = 2$ and $p = \infty$.

4.6. Implementation details

The implementations of the adversarial attacks, including the gradient estimator using the natural evolution strategies, were provided by the Foolbox library [57]. The implementation of the Venn-ABERS predictor was provided by Toccaceli [58]. The different data splits we perform for these experiments are illustrated schematically in Figure 4. Almost all neural networks were trained for 50 epochs using the Adam optimizer [49] with default parameters in the Keras framework [59] using the TensorFlow backend [60]. The exception is the CNN for the cats vs dogs task, which was trained for 100 epochs. No regularization or data augmentation schemes were used; the only preprocessing we perform is a normalization of the input pixels to the [0,1] range as well as resizing the images in one of the tasks. Descriptions of all the CNN architectures can be found in Appendix A.

4.7. Data sets

Here, we detail the different data sets used for our experiments. Note that these are all *binary* classification problems, since the IVAP construction as formulated by Vovk et al. [11] only works for binary tasks. Extending the IVAP to multiclass problems has been discussed, for example, in Manokhin [44]; using a multiclass IVAP in adversarial defense will be explored in future work.

Zeroes vs ones. Our first task is a binary classification problem based on the MNIST data set [61]. We take the original MNIST data set and filter out only the images displaying either a zero or a one. The pixel values are normalized to lie in the interval [0,1]. We then run our experiments as described above, using a convolutional neural network (CNN) as the scoring classifier for Algorithm 3.

Cats vs dogs. The second task we consider is the classification of cats and dogs in the Asirra data set [62]. This data set consists of 25,000 JPEG color images where half contain dogs and half contain cats. Again we train a CNN as the scoring classifier for Algorithm 3. In the original collection, not all images were of the same size. In our experiments, we resized all images to 64×64 pixels and normalized the pixel values to [0,1] to facilitate processing by our machine learning pipeline.

T-shirts vs trousers. The third task is classifying whether a 28×28 grayscale image contains a picture of a T-shirt or a pair of trousers. Similarly to the MNIST zeroes vs ones task, we take the Fashion-MNIST data set [63] and filter out the pictures of T-shirts and trousers. Again, the pixel values are normalized to [0,1] and a CNN is used for this classification problem.

Airplanes vs automobiles. Our fourth task is based on the CIFAR-10 data set [30], which consists of 60,000 RGB images 32×32 pixels in size. We filter out the images of airplanes and automobiles and train a CNN to distinguish these two classes.

4.8. Results and discussion

The baseline accuracies of the unprotected CNNs are given in Table 1 for the different tasks. We can see that they perform reasonably accurately on clean test data, but as expected the adversarial accuracy is very low. Table 2 gives the clean accuracies of the IVAPs for each of the tasks. From these results, we can see that the clean accuracy of the protected model invariably suffers from the addition of the IVAP. However, in almost all tasks, the false positive and false negative rates are relatively low. Our detectors are therefore capable of accepting correct predictions and rejecting mistaken ones from the underlying model on clean data. The exception is the cats vs dogs task, where the false positive rate is unusually high at 51.15%. However, we believe these results might be improved by using a better underlying CNN. In our initial experiments, we used the same CNN for this task as we did for airplanes vs automobiles and trained it for only 50 epochs. Using a more complex CNN and training it longer significantly improved the results, so we are optimistic that the results we obtained can be improved even further by adapting the underlying CNN and training regime.

The results presented here are observations from a single run of our experiments. Multiple runs of the experiments did not produce significantly different results.

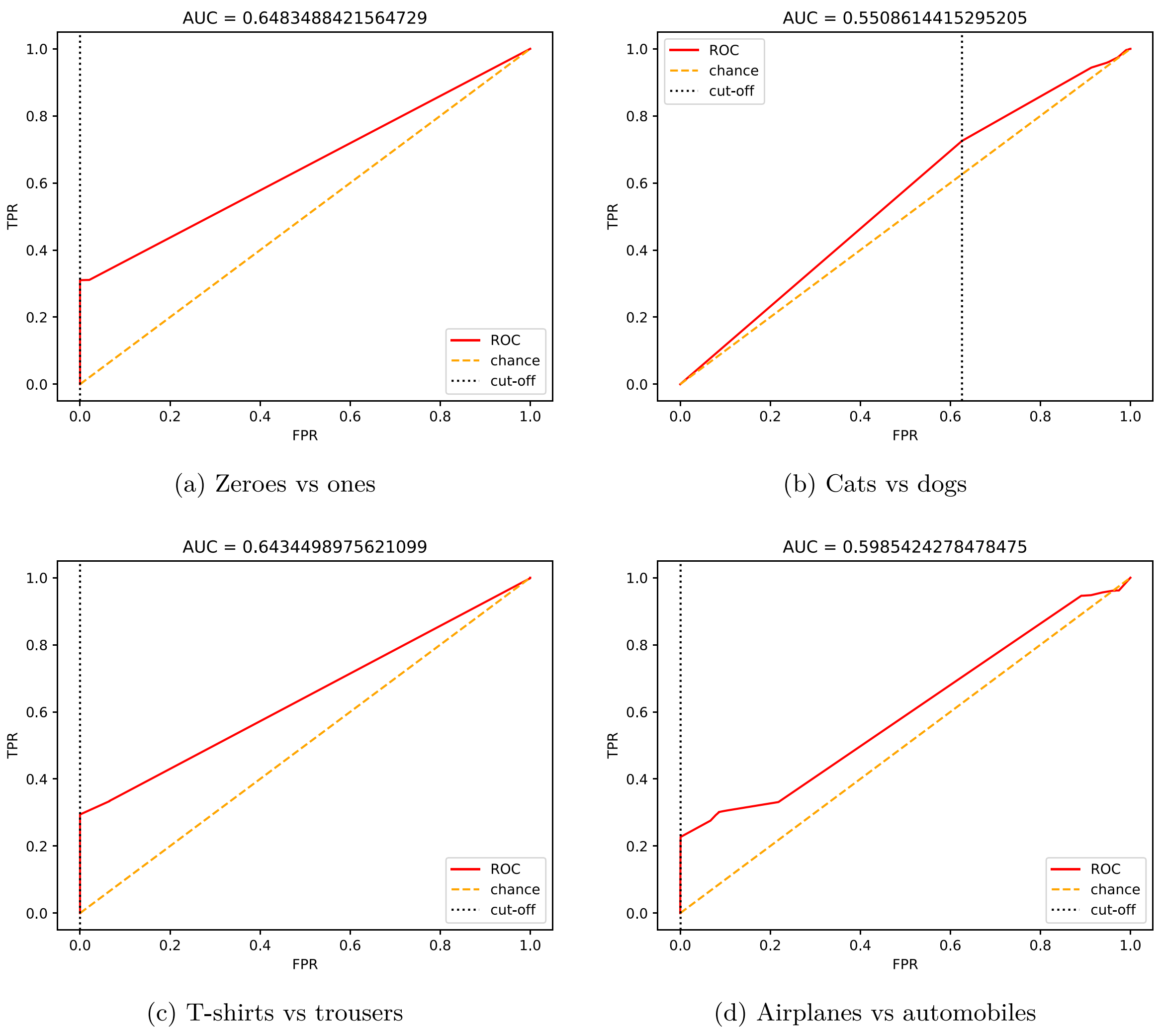


Fig. 5. The ROC curves for the different detectors. The orange dashed line is the random chance line. The red solid line is the ROC curve for our detector. The dotted black line is the cut-off corresponding to the optimal value of β, which is defined here as the value that maximizes Youden's index.

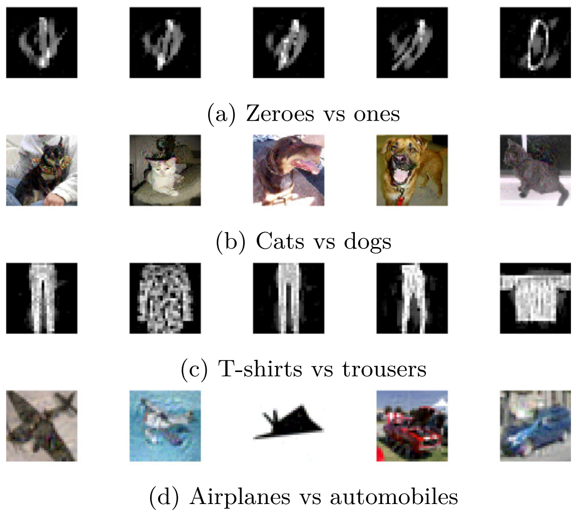


Fig. 6. Selections of adversarial examples which can fool our detectors, generated by the custom ℓ_2 white-box attack.

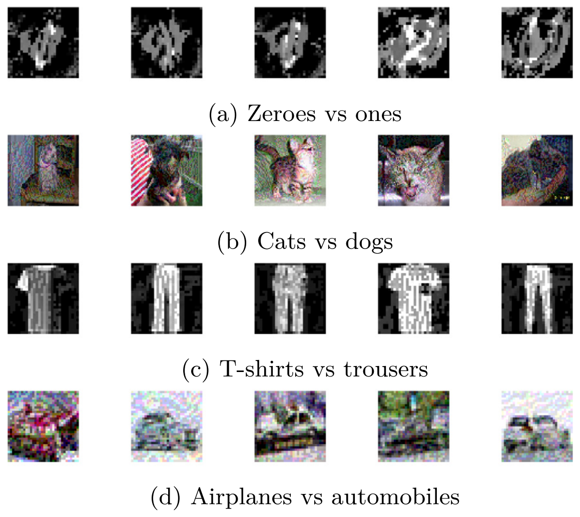


Fig. 7. Selections of adversarial examples which can fool our detectors, generated by the custom ℓ_∞ white-box attack.

Robustness to existing attacks. The first question we would like to answer when evaluating any novel adversarial defense is whether it can defend against existing attacks that were not adapted specifically to fool it. This is a bare minimum of strength one desires: if existing attacks can already break the defense, then it is useless. In Table 2, we evaluate our IVAP defense against the

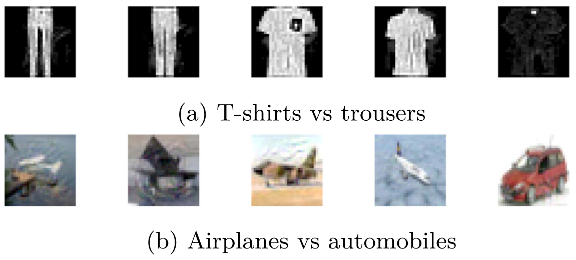


Fig. 8. Selections of adversarial examples which can fool our ablated detectors, generated by the custom ℓ_2 white-box attack.

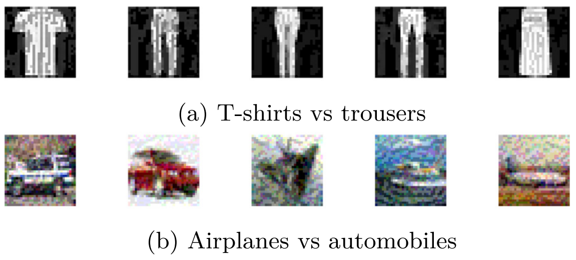


Fig. 9. Selections of adversarial examples which can fool our ablated detectors, generated by the custom ℓ_∞ white-box attack.

suite of attacks described in Section 4.1. The results are shown for the *Adversarial* and *Adapted* scenarios. When we transfer adversarials generated for the underlying model to the IVAP, the accuracy always degrades significantly. However, it is important to note that all unprotected CNNs have very low accuracy on these samples. As such, in almost all cases, the IVAP has a very high true negative rate as most predictions are in error. The exception is the cats vs dogs task, where the true negative rate is rather low on the transferred adversarials. The accuracy is still much higher than the unprotected model, though (52.56% vs 1.8%), so the IVAP did significantly improve the robustness against this transfer attack. The adversarials generated using existing attacks adapted to the presence of the IVAP also degrade the accuracy, but not as severely as the transfer attack. True negative rates remain high across all tasks and the level of accuracy is, at the very least, much better than without IVAP protection.

Robustness to novel attacks. To more thoroughly evaluate the strength of our defense, it is necessary to also test it against an adaptive adversary utilizing an attack that is specifically designed to circumvent our defense. For this reason, Table 2 also shows two rows of test results for each task, where we run our custom white-box attack for both the ℓ_2 and ℓ_∞ norms. At first glance, it appears as though our attack is completely successful and fully breaks our defense, yielding 0% accuracy. To investigate this further, Figs. 6 and 7 show random selections of five adversarial examples that managed to fool the different detectors, generated by our white-box attack. Empirical cumulative distributions of the distortion levels introduced by our white-box attack are shown in Figs. 10 and 11. Visually inspecting the generated adversarials and looking at the distributions of the distortion levels reveals cause for optimism: although some perturbations are still visually imperceptible, there is a significant increase in the overall distortion required to fool the detectors. This is especially the case for the ℓ_∞ adversarials, where almost all perturbations are obvious. Even the ℓ_2 adversarials are generally very unsubtle except for the ones generated on the cats vs dogs task. We believe this is due to the fact that the Asirra images are larger than the CIFAR-10 images (64×64 vs 32×32), so that the ℓ_2 perturbations can be more "spread out" across the pixels. Regardless, if one compares the adversarial images to the originals, the perturbations are still obvious because the adversarials are unusually grainy and blurred.

Sensitivity of hyperparameters. For the experiments carried out in Table 2, the IVAP was only exposed to adversarial attacks which it had already seen: the rejection threshold β was tuned on samples perturbed using the same adversarial attacks as it was then tested against. Fig. 5 shows the ROC curves for the different detectors along with the optimal thresholds. An important question that arises now is how well the IVAP fares if we test it against attacks that differ from the ones used to tune its hyperparameters, since in practice it is doubtful that we will be able to anticipate what attacks an adversary may use. Moreover, even if we can anticipate this, the set of possible attacks may be so large as to make it computationally infeasible to use them all. Therefore, we also test the performance of the IVAP when β is tuned on adversarials generated only by the DeepFool attack instead of the full suite of attacks. We obtained identical values of β for the Zeroes vs ones and Cats vs dogs tasks. The results for the other tasks where β was different are shown in Table 3. The conclusions are similar: the accuracy degrades both on clean and adversarial data, but clean accuracy is still high and true negative rates are high on adversarial data. The custom white-box attack again appears highly successful. Figs. 12 and 13 plot the empirical cumulative distributions of the adversarial distortion levels for the ablated detectors and Figs. 8 and 9 show selections of custom white-box adversarials which fooled the ablated detectors. These results show that robustness to our ℓ_2 white-box attack is significantly lower compared to the non-ablated detectors. The ℓ_∞-bounded perturbations are still very noticable, however. As an illustrative example, Fig. 14 shows histograms of the differences $p_1 - p_0$ between the upper and lower probabilities for clean and DeepFool adversarial examples on the zeroes vs ones task. Although there is a small amount of overlap, the vast majority of adversarial examples have a much higher difference than clean samples: clean data has a difference close to 0%, whereas adversarials are almost always close to 50%. The tuned model threshold is placed so that the majority of clean samples are still allowed through but virtually all adversarials are rejected.

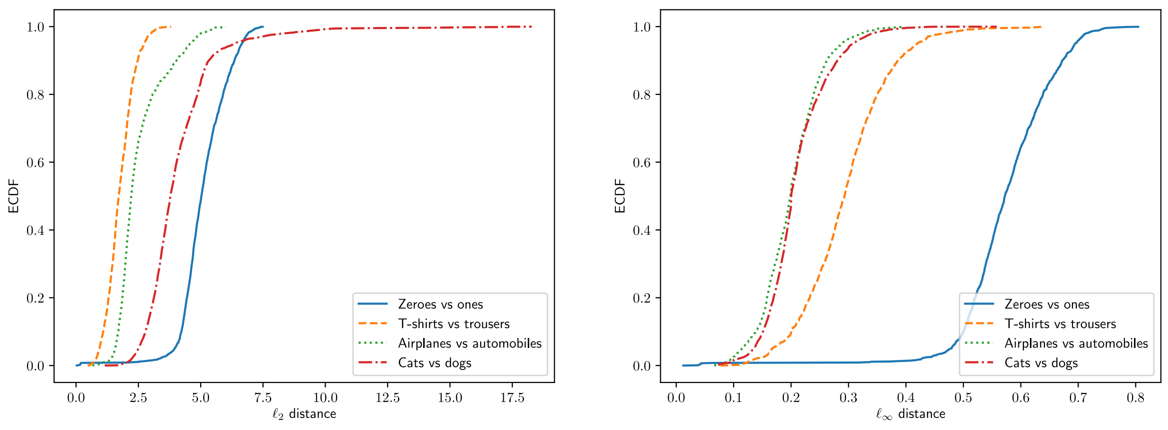
5. Comparison with other methods

In this section, we compare our IVAP method to the Madry defense [27] which, to the best of our knowledge, is the strongest state-of-the-art defense at the time of this writing. Specifically, we apply the Madry defense to each of the CNNs used in the tasks from Section 4 and compare the accuracies of the different methods. We also perform transfer attacks where we try to fool our IVAP using adversarials generated for the Madry defense and vice versa.

The Madry defense is a form of adversarial training where, at each iteration of the learning algorithm, the ℓ_∞ projected gradient descent attack is used to perturb all samples in the current minibatch[4]. The model is then updated based on this perturbed batch instead of the original. To facilitate fair comparison, we do not use the pre-trained models and weights published by Madry et al. for the MNIST and CIFAR-10 data sets, since these are meant for the full 10-class problems whereas our defense only considers a simpler binary subproblem. Rather, we modified the network architectures for these tasks so they correspond to our own CNNs from Section 4 and trained them according to the procedure outlined by Madry et al. [27].

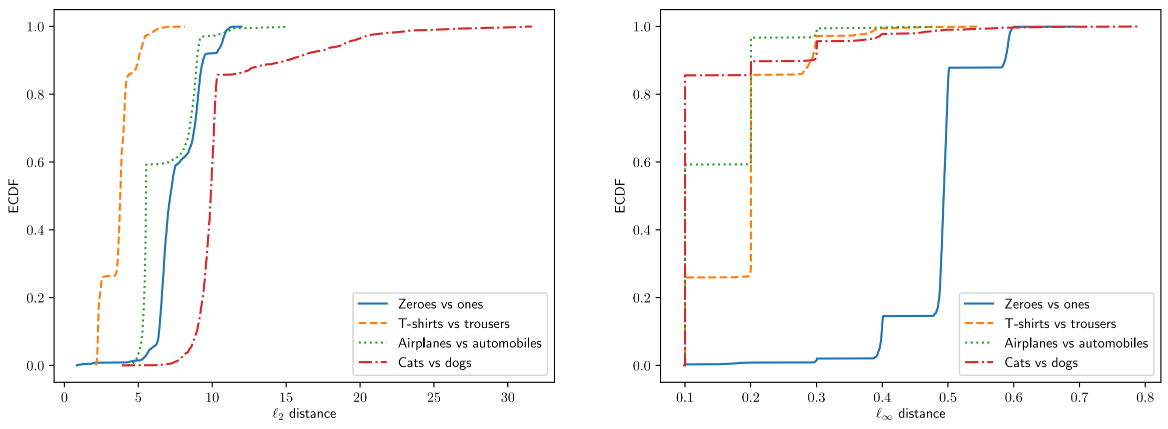
Table 4 shows the results we obtained with the Madry defense as well as the parameter settings we used for the PGD attack. The

[4] Madry et al. stress that the ℓ_∞ variant be used and not, e.g., an ℓ_2 one. We follow their advice and perform adversarial training exclusively with the ℓ_∞ version. The implementation we used was taken directly from https://github.com/MadryLab/mnist_challenge.



(a) ℓ_2 (b) ℓ_∞

Fig. 10. Empirical cumulative distributions of the adversarial distortion produced by our ℓ_2 white-box attack. The distance metrics are ℓ_2 (left) and ℓ_∞ (right).



(a) ℓ_2 (b) ℓ_∞

Fig. 11. Empirical cumulative distributions of the adversarial distortion produced by our ℓ_∞ white-box attack. The distance metrics are ℓ_2 (left) and ℓ_∞ (right).

Table 3
Summary of performance indicators for the ablated detectors on the different tasks.

Task	Data	Size	Accuracy	TPR	TNR	FPR	FNR
T-shirts vs trousers	Clean	1,600	97.81%	97.89%	94.87%	5.13%	2.11%
	Adversarial	7,076	49.31%	3.81%	93.02%	6.98%	96.19%
	Adapted	5,901	56.8%	0.0%	56.93%	43.07%	100.0%
	Custom ℓ_2	1,600	0.0%	0.0%	0.0%	100.0%	100.0%
	Custom ℓ_∞	1,600	0.0%	0.0%	0.0%	100.0%	100.0%
Airplanes vs automobiles	Clean	1,600	93.62%	95.39%	60.49%	39.51%	4.61%
	Adversarial	8,256	52.63%	0.0%	90.52%	9.48%	100.0%
	Adapted	6,580	69.18%	0.0%	69.73%	30.27%	100.0%
	Custom ℓ_2	1,600	0.0%	0.0%	0.0%	100.0%	100.0%
	Custom ℓ_∞	1,598	0.0%	0.0%	0.0%	100.0%	100.0%

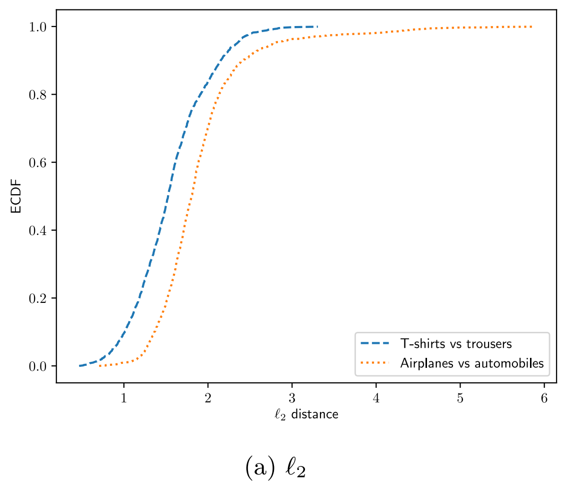
Table 4
Summary of parameters and performance indicators for the Madry defense on the machine learning tasks we considered. Step sizes were always set to 0.01 and random start was used each time.

Task	Clean accuracy	Adversarial accuracy	ε	Steps
Zeroes vs ones	99.88%	97.1%	0.3	40
Cats vs dogs	54.94%	3.5%	0.3	10
T-shirts vs trousers	97.12%	85%	0.3	40
Airplanes vs automobiles	82.06%	29.19%	0.3	10

Table 5
Results of the Madry-to-IVAP transfer attack, where adversarial examples generated for the Madry defense were tested against our IVAPs.

Task	Accuracy	TPR	TNR	FPR	FNR
Zeroes vs ones	36.05%	30.91%	100.0%	0.0%	69.09%
Cats vs dogs	61.64%	65.08%	52.3%	47.7%	34.92%
T-shirts vs trousers	72.75%	71.02%	84.47%	15.53%	28.98%
Airplanes vs automobiles	45.81%	34.53%	95.92%	4.08%	65.47%

defense performs very well on the zeroes vs ones and T-shirts vs trousers tasks. However, on airplanes vs automobiles the adversarial accuracy is low and on the cats vs dogs task both the clean and adversarial accuracies are low. In fact, for the latter task, the adversarial accuracy is almost the same as if no defense was applied at all.



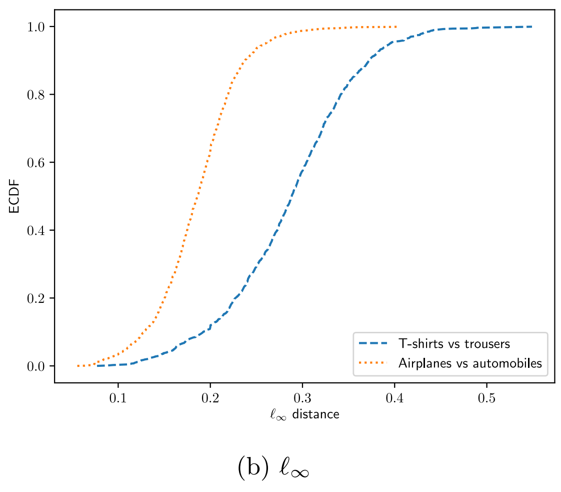
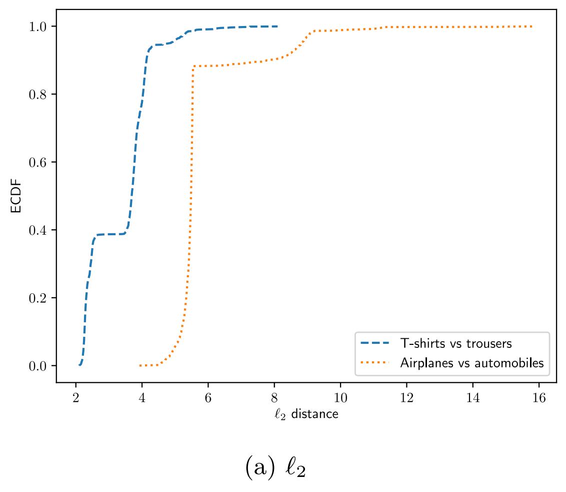


Fig. 12. Empirical cumulative distributions of the adversarial distortion produced by our ℓ_2 white-box attack on the ablated detectors. The distance metrics are ℓ_2 (left) and ℓ_∞ (right).



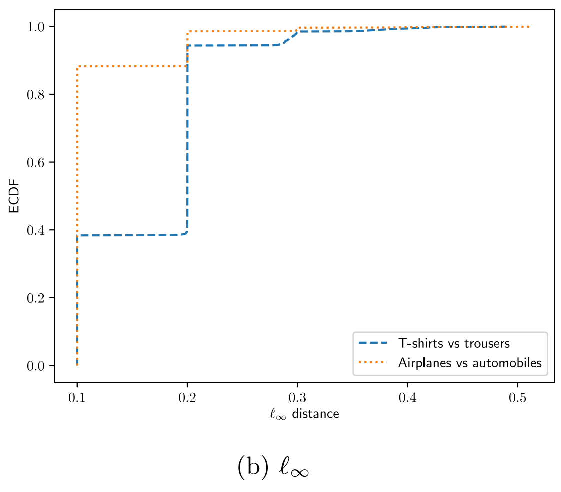


Fig. 13. Empirical cumulative distributions of the adversarial distortion produced by our ℓ_∞ white-box attack on the ablated detectors. The distance metrics are ℓ_2 (left) and ℓ_∞ (right).

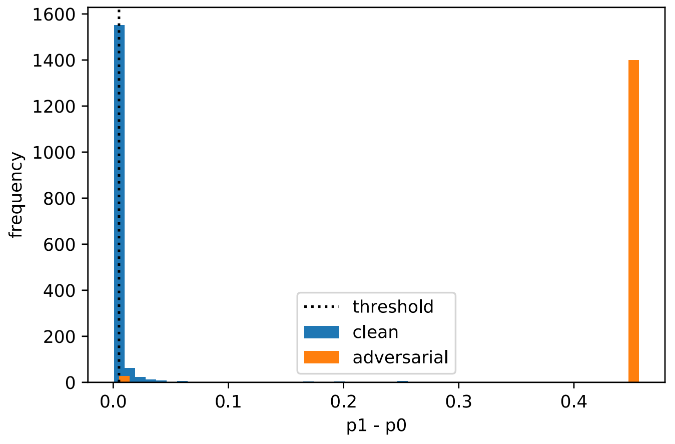


Fig. 14. Histograms of the differences $p_1 - p_0$ for clean and adversarial data along with the tuned model threshold β. We used the DeepFool attack here to generate adversarials for the zeroes vs ones model.

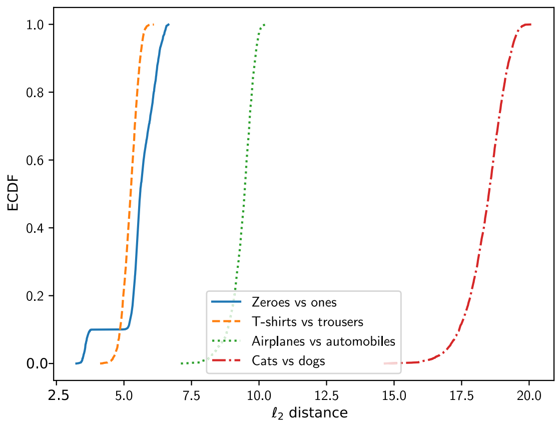
In Table 6 we transfer the white-box adversarials for our defense to the Madry defense. The Madry defense appears quite robust to our IVAP adversarials for T-shirts vs trousers and airplanes vs automobiles, but much less so on zeroes vs ones. The accuracy on the adversarials for the cats vs dogs task is close to the clean accuracy, so these appear to have little effect on the Madry defense.

Table 6
Results of the IVAP-to-Madry transfer attack. Here, adversarial examples generated for the IVAP defense by our custom white-box attack are transferred to the Madry defense. We test adversarials generated by both the ℓ_2 and ℓ_∞ variants of our attack.

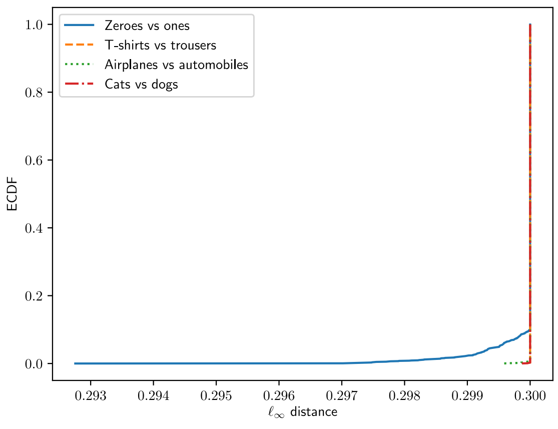
Task	Accuracy (ℓ_2)	Accuracy (ℓ_∞)
Zeroes vs ones	46.28%	66.78%
Cats vs dogs	55.26%	55.48%
T-shirts vs trousers	94.94%	93.88%
Airplanes vs automobiles	79.94%	78.38%

Table 5 shows what happens when we transfer adversarials for the Madry defense to our IVAP detectors. We observe that for the zeroes vs ones and airplanes vs automobiles, the accuracy drops below 50%. However, false positive rates on these adversarials are very low, so the detectors successfully reject many incorrect predictions. On the other tasks, accuracy remains relatively high. The false positive rate is rather high for the cats vs dogs task, but low for T-shirts vs trousers.

Fig. 15 plots the empirical distributions of the adversarial distortion levels of the Madry adversarials, both for the ℓ_2 and ℓ_∞

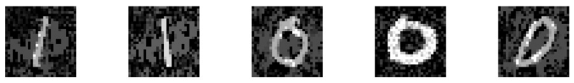


(a) ℓ_2



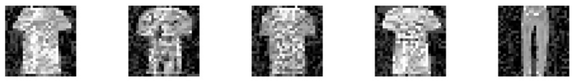
(b) ℓ_∞

Fig. 15. Empirical cumulative distribution curves for the adversarial distortion of the Madry adversarials.

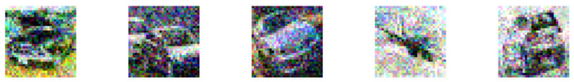


(a) Zeroes vs ones

(b) Cats vs dogs



(c) T-shirts vs trousers



(d) Airplanes vs automobiles

Fig. 16. Selections of adversarial examples which can fool the Madry defense, generated by the ℓ_∞ PGD attack.

distances. Comparing this figure with Figs. 10 and 11, we see that our custom white-box adversarials for the IVAP defense generally require higher ℓ_2 and ℓ_∞ levels of distortion than the Madry adversarials. Fig. 16 plots selections of adversarials generated for the Madry defenses by the ℓ_∞ PGD attack. The visual levels of distortion appear comparable to our defense (for the ℓ_∞ variant of our attack), except in the case of the cats vs dogs task where the perturbations for the Madry defense seem much larger. It is important to note, however, that the performance of the Madry defense on this task is close to random guessing, whereas the IVAP still obtains over 70% clean accuracy.

We are lead to the conclusion that our defense in fact achieves higher robustness on these tasks than the Madry defense. Moreover, our defense appears to scale much better: adversarially training a deep CNN with ℓ_∞ projected gradient descent quickly becomes computationally intractable as the data increase in dimensionality and the CNN increases in complexity. On the other hand, our IVAP defense can take an existing CNN (trained using efficient, standard methods) and quickly augment it with uncertainty estimates based on a calibration set and a data set of adversarials for the underlying model. The resulting algorithm appears to require higher levels of distortion than the Madry defense in order to be fooled effectively and it seems to achieve higher clean and adversarial accuracy on more realistic tasks. Finally, we note that our method introduces only a single extra scalar parameter, the threshold β, which can be easily manually tuned if desired. It is also possible to tune β based on other objectives than Youden's index which was used here. On the other hand, the Madry defense requires a potentially lengthy adversarial training procedure whenever the defense's parameters are modified.

6. Conclusion

We have proposed using inductive Venn-ABERS predictors (IVAPs) to protect machine learning models against adversarial manipulation of input data. Our defense uses the width of the uncertainty interval produced by the IVAP as a measure of confidence in the prediction of the underlying model, where the prediction is rejected in case this interval is too wide. The acceptable width is a hyperparameter of the algorithm which can be estimated using a validation set. The resulting algorithm is no longer vulnerable to the original adversarial examples that fooled the unprotected model and attacks which take the detector into account have very limited success. We do not believe this success to be due to gradient masking, as the defense remains robust even when subjected to attacks that do not suffer from this phenomenon. Our method appears to be competitive to the defense proposed by Madry et al. [27], which is state-of-the-art at the time of this writing.

The algorithm proposed here is limited to binary classification problems. However, there exist methods to extend the IVAP to the multiclass setting [44]. Our future work will focus on using these techniques to generalize our detector to multiclass classification problems.

Of course, the mere fact that our own white-box attack fails against the IVAP defense does not constitute hard proof that our method is successful. We therefore invite the community to scrutinize our defense and to attempt to develop stronger attacks against it. Our code is available at https://github.com/saeyslab/binary-ivap. The performance of the IVAP defense is still not ideal at this stage since clean accuracy is noticeably reduced. However, we believe these findings represent a significant step forward. As such, we suggest that the community further look into methods from the field of conformal prediction in order to achieve adversarial robustness at scale. To our knowledge, we are the first to apply these methods to this problem, although the idea has been mentioned elsewhere already [64, Section 9.3, p133].

Declaration of Competing Interest

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

Acknowledgements

We gratefully acknowledge the support of the NVIDIA corporation who provided us with a Titan Xp GPU to conduct our experiments under the GPU Grant Program. Jonathan Peck is sponsored by a Ph.D. fellowship of the Research Foundation - Flanders (FWO). Yvan Saeys is an ISAC Marylou Ingram Scholar.

Appendix A. Description of the Neural Networks

In this section, we give detailed descriptions of the CNNs used in our experiments. These descriptions are Python code for the Keras framework. Note that we assume the following imports have been loaded:

```
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D, MaxPooling2D, Activation
```

A1. Zeroes vs ones

```
model = Sequential()
model.add(Conv2D(32, kernel_size=(5, 5),
                activation='relu',
                input_shape=(28, 28, 1)))
model.add(Conv2D(64, (5, 5), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(1024, activation='relu'))
model.add(Dense(2))
model.add(Activation('softmax'))
```

A2. Cats vs dogs

```
model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3),
                activation='relu',
                input_shape=(64, 64, 3)))
model.add(Conv2D(32, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(.25))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(.25))
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(.25))
model.add(Flatten())
model.add(Dense(512, activation='relu'))
model.add(Dropout(.5))
model.add(Dense(2))
model.add(Activation('softmax'))
```

A3. T-shirts vs trousers

```
model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3),
                activation='relu',
                input_shape=(28, 28, 1)))
model.add(Conv2D(32, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(.25))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(.25))
model.add(Flatten())
model.add(Dense(512, activation='relu'))
model.add(Dropout(.5))
model.add(Dense(2))
model.add(Activation('softmax'))
```

A4. Airplanes vs automobiles

```
model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3),
                activation='relu',
                input_shape=(32, 32, 3)))
model.add(Conv2D(32, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(.25))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(.25))
model.add(Flatten())
model.add(Dense(512, activation='relu'))
model.add(Dropout(.5))
model.add(Dense(2))
model.add(Activation('softmax'))
```

References

[1] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (7553) (2015) 436.
[2] J. Schmidhuber, Deep learning in neural networks: an overview, Neural Netw. 61 (2015) 85–117.
[3] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, L. Fei-Fei, Imagenet: a large-scale hierarchical image database, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Ieee, 2009, pp. 248–255.
[4] J. Platt, Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods, Adv. Large Margin Classif. 10 (3) (1999) 61–74.
[5] Y. Gal, Uncertainty in Deep Learning, University of Cambridge (2016).
[6] I.J. Goodfellow, J. Shlens, C. Szegedy, Explaining and harnessing adversarial examples, CoRR, abs/1412.6572 (2015).
[7] A. Nguyen, J. Yosinski, J. Clune, Deep neural networks are easily fooled: High confidence predictions for unrecognizable images, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 427–436.
[8] G. Shafer, A. Gammerman, V. Vovk, Algorithmic Learning in a Random World, Springer, 2005.
[9] A. Gammerman, V. Vovk, Hedging predictions in machine learning, Comput. J. 50 (2) (2007) 151–163.
[10] G. Shafer, V. Vovk, A tutorial on conformal prediction, J. Mach. Learn. Res. 9 (Mar) (2008) 371–421.
[11] V. Vovk, I. Petej, V. Fedorova, Large-scale probabilistic predictors with and without guarantees of validity, in: Advances in Neural Information Processing Systems, 2015, pp. 892–900.
[12] N. Dalvi, P. Domingos, Mausam, S. Sanghai, D. Verma, Adversarial classification, in: Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2004, pp. 99–108.
[13] M. Barreno, B. Nelson, R. Sears, A.D. Joseph, J.D. Tygar, Can machine learning be secure? in: Proceedings of the ACM Symposium on Information, Computer and Communications Security, ACM, 2006, pp. 16–25.
[14] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, R. Fergus, Intriguing properties of neural networks, arXiv:1312.6199 (2013).
[15] B. Biggio, F. Roli, Wild patterns: Ten years after the rise of adversarial machine learning, Pattern Recognit. 84 (2018) 317–331.
[16] N. Carlini, D. Wagner, Towards evaluating the robustness of neural networks, in: Proceedings of the IEEE Symposium on Security and Privacy (SP), IEEE, 2017, pp. 39–57.
[17] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z.B. Celik, A. Swami, Practical black-box attacks against machine learning, in: Proceedings of the ACM Asia Conference on Computer and Communications Security, in: ASIA CCS '17, ACM, 2017, pp. 506–519.
[18] Z. Gong, W. Wang, W.-S. Ku, Adversarial and clean data are not twins, arXiv:1704.04960 (2017).

[19] D. Hendrycks, K. Gimpel, Early methods for detecting adversarial images, arXiv:1608.00530 (2016).
[20] K. Grosse, P. Manoharan, N. Papernot, M. Backes, P. McDaniel, On the (statistical) detection of adversarial examples, arXiv:1702.06280 (2017).
[21] X. Li, F. Li, Adversarial examples detection in deep networks with convolutional filter statistics., in: Proceedings of the International Conference on Computer Vision (ICCV), 2017, pp. 5775–5783.
[22] S. Gu, L. Rigazio, Towards deep neural network architectures robust to adversarial examples, arXiv:1412.5068 (2014).
[23] F. Liao, M. Liang, Y. Dong, T. Pang, J. Zhu, X. Hu, Defense against adversarial attacks using high-level representation guided denoiser, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 1778–1787.
[24] A. Graese, A. Rozsa, T.E. Boult, Assessing threat of adversarial examples on deep neural networks, in: Proceedings of the 15th IEEE International Conference on Machine Learning and Applications (ICMLA), IEEE, 2016, pp. 69–74.
[25] M. Osadchy, J. Hernandez-Castro, S. Gibson, O. Dunkelman, D. Pérez-Cabo, No bot expects the DeepCAPTCHA! Introducing immutable adversarial examples, with applications to CAPTCHA generation, IEEE Trans. Inf. Forens. Secur. 12 (11) (2017) 2640–2653.
[26] M. Cisse, P. Bojanowski, E. Grave, Y. Dauphin, N. Usunier, Parseval networks: improving robustness to adversarial examples, arXiv:1704.08847 (2017).
[27] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, A. Vladu, Towards deep learning models resistant to adversarial attacks, in: Proceedings of the International Conference on Learning Representations, 2018.
[28] F. Tramèr, A. Kurakin, N. Papernot, I. Goodfellow, D. Boneh, P. McDaniel, Ensemble adversarial training: attacks and defenses, arXiv:1705.07204 (2017).
[29] I. Goodfellow, Defense against the dark arts: an overview of adversarial example security research and future research directions, arXiv:1806.04169 (2018).
[30] A. Krizhevsky, G. Hinton, Learning multiple layers of features from tiny images, Technical Report, Citeseer, 2009.
[31] B. Graham, Fractional max-pooling, arXiv:1412.6071 (2014).
[32] A. Athalye, N. Carlini, D. Wagner, Obfuscated gradients give a false sense of security: circumventing defenses to adversarial examples, arXiv:1802.00420 (2018).
[33] W. Brendel, J. Rauber, M. Bethge, Decision-based adversarial attacks: reliable attacks against black-box machine learning models, arXiv:1712.04248 (2017).
[34] A. Ilyas, L. Engstrom, A. Athalye, J. Lin, Black-box adversarial attacks with limited queries and information, arXiv:1804.08598 (2018).
[35] L. Smith, Y. Gal, Understanding measures of uncertainty for adversarial example detection, arXiv:1803.08533 (2018).
[36] A. Rawat, M. Wistuba, M.-I. Nicolae, Adversarial phenomenon in the eyes of Bayesian deep learning, arXiv:1711.08244 (2017).
[37] R. Feinman, R.R. Curtin, S. Shintre, A.B. Gardner, Detecting adversarial samples from artifacts, arXiv:1703.00410 (2017).
[38] Y. Li, Y. Gal, Dropout inference in Bayesian neural networks with alpha-divergences, arXiv:1703.02914 (2017).
[39] C. Zhang, J. Butepage, H. Kjellstrom, S. Mandt, Advances in variational inference, arXiv:1711.05597 (2017).
[40] D.A. Fraser, Is Bayes posterior just quick and dirty confidence? Stat. Sci. 26 (3) (2011) 299–316.
[41] N. Carlini, D. Wagner, Adversarial examples are not easily detected: bypassing ten detection methods, in: Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security, ACM, 2017, pp. 3–14.
[42] Y.S. Jonathan Peck Bart Goossens, Detecting adversarial examples with inductive Venn-ABERS predictors, in: Proceedings of the European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, 2019, pp. 143–148.
[43] L. Wasserman, Frasian inference, Stat. Sci. (2011) 322–325.
[44] V. Manokhin, Multi-class probabilistic classification using inductive and cross Venn–Abers predictors, in: Conformal and Probabilistic Prediction and Applications, 2017, pp. 228–240.
[45] R Core Team, R: A Language and Environment for Statistical Computing, R Foundation for Statistical Computing, Vienna, Austria, 2015.
[46] H. Tuy, Convex Analysis and Global Optimization, Springer, 1998.
[47] B. Klaus, K. Strimmer, FDRTOOL: Estimation of (Local) False Discovery Rates and Higher Criticism, 2015. R package version 1.2.15.
[48] W.J. Youden, Index for rating diagnostic tests, Cancer 3 (1) (1950) 32–35.
[49] D.P. Kingma, J. Ba, Adam: a method for stochastic optimization, arXiv:1412.6980 (2014).
[50] S.-M. Moosavi-Dezfooli, A. Fawzi, P. Frossard, DeepFool: a simple and accurate method to fool deep neural networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 2574–2582.
[51] N. Narodytska, S.P. Kasiviswanathan, Simple black-box adversarial perturbations for deep networks, arXiv:1612.06299 (2016).
[52] J. Su, D.V. Vargas, S. Kouichi, One pixel attack for fooling deep neural networks, arXiv:1710.08864 (2017).
[53] U. Jang, X. Wu, S. Jha, Objective metrics and gradient descent algorithms for adversarial examples in machine learning, in: Proceedings of the 33rd Annual Computer Security Applications Conference, ACM, 2017, pp. 262–277.
[54] Y. Dong, F. Liao, T. Pang, H. Su, J. Zhu, X. Hu, J. Li, Boosting adversarial attacks with momentum, (2018).
[55] D. Wierstra, T. Schaul, T. Glasmachers, Y. Sun, J. Peters, J. Schmidhuber, Natural evolution strategies, J. Mach. Learn. Res. 15 (2014) 949–980.
[56] A. Kurakin, I. Goodfellow, S. Bengio, Y. Dong, F. Liao, M. Liang, T. Pang, J. Zhu, X. Hu, C. Xie, J. Wang, Z. Zhang, Z. Ren, A. Yuille, S. Huang, Y. Zhao, Y. Zhao, Z. Han, J. Long, Y. Berdibekov, T. Akiba, S. Tokui, M. Abe, Adversarial attacks and defences competition, arXiv:1804.00097 (2018).
[57] J. Rauber, W. Brendel, M. Bethge, Foolbox: A Python toolbox to benchmark the robustness of machine learning models, arXiv:1707.04131 (2017).
[58] P. Toccaceli, Venn-ABERS Predictor, 2017, (https://github.com/ptocca/VennABERS). Accessed: 25 September 2018.
[59] F. Chollet, et al., Keras, 2015, (https://keras.io).
[60] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G.S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, X. Zheng, TensorFlow: Large-scale machine learning on heterogeneous systems, 2015, Software available from tensorflow.org.
[61] Y. LeCun, The MNIST database of handwritten digits, 1998, (http://yann.lecun.com/exdb/mnist/). Accessed: 25 September 2018.
[62] J. Elson, J.J. Douceur, J. Howell, J. Saul, Asirra: a CAPTCHA that exploits interest-aligned manual image categorization, in: Proceedings of the 14th ACM Conference on Computer and Communications Security (CCS), ACM, 2007.
[63] H. Xiao, K. Rasul, R. Vollgraf, Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms, arXiv:1708.07747(2017).
[64] Y. Vorobeychik, M. Kantarcioglu, Adversarial machine learning, Synthes. Lect. Artif. Intell. Mach. Learni. 12 (3) (2018) 1–169.

Jonathan Peck received the B.Sc. degree in Computer Science and M.Sc. in Mathematical Informatics at Ghent University, Belgium, in 2015 and 2017 respectively. He is currently pursuing a Ph.D. at Ghent University, sponsored by a fellowship of the Research Foundation Flanders (FWO). His research focuses on improving the robustness of machine learning models to adversarial manipulations.

Bart Goossens is a professor at Ghent University (department TELIN/IPI/imec), in the field of digital image processing. He obtained the M.Sc. degree in Computer Science in 2006 and a Ph.D. in Engineering in 2010 at Ghent University. His research focuses on digital restoration of images and video, noise modeling and estimation, super-resolution and medical image reconstruction.

Yvan Saeys graduated as a computer scientist from Ghent University in 2000, and obtained his PhD in computer science at the Bioinformatics group of the Flemish Institute for Biotechnology (VIB) and Ghent University. He established the DAMBI research group at the Inflammation Research Centre (IRC), focusing on the development and application of new data mining and machine learning techniques for biomedicine. His research focuses on the interplay between computer science and biology and medicine, particularly the tailored design of state-of-the-art computational techniques for specific biological or medical questions.