The Odds are Odd: A Statistical Test for Detecting Adversarial Examples

Kevin Roth [*1] **Yannic Kilcher** [*1] **Thomas Hofmann** [1]

Abstract

We investigate conditions under which test statistics exist that can reliably detect examples, which have been adversarially manipulated in a white-box attack. These statistics can be easily computed and calibrated by randomly corrupting inputs. They exploit certain anomalies that adversarial attacks introduce, in particular if they follow the paradigm of choosing perturbations optimally under p-norm constraints. Access to the log-odds is the only requirement to defend models. We justify our approach empirically, but also provide conditions under which detectability via the suggested test statistics is guaranteed to be effective. In our experiments, we show that it is even possible to correct test time predictions for adversarial attacks with high accuracy.

1. Introduction

Deep neural networks have been used with great success for perceptual tasks such as image classification (Simonyan & Zisserman, 2014; LeCun et al., 2015) or speech recognition (Hinton et al., 2012). While they are known to be robust to random noise, it has been shown that the accuracy of deep nets can dramatically deteriorate in the face of so-called adversarial examples (Biggio et al., 2013; Szegedy et al., 2013; Goodfellow et al., 2014), i.e. small perturbations of the input signal, often imperceptible to humans, that are sufficient to induce large changes in the model output.

A plethora of methods have been proposed to find adversarial examples (Szegedy et al., 2013; Goodfellow et al., 2014; Kurakin et al., 2016; Moosavi Dezfooli et al., 2016; Sabour et al., 2015). These often transfer across different architectures, enabling black-box attacks even for inaccessible models (Papernot et al., 2016; Kilcher & Hofmann, 2017; Tramèr et al., 2017). This apparent vulnerability is worrisome as deep nets start to proliferate in the real-world, including in safety-critical deployments.

The most direct and popular strategy of robustification is to use adversarial examples as data augmentation during training (Goodfellow et al., 2014; Kurakin et al., 2016; Madry et al., 2017), which improves robustness against specific attacks, yet does not address vulnerability to more cleverly designed counter-attacks (Athalye et al., 2018; Carlini & Wagner, 2017a). This raises the question of whether one can protect models with regard to a wider range of possible adversarial perturbations.

A different strategy of defense is to detect whether or not the input has been perturbed, by detecting characteristic regularities either in the adversarial perturbations themselves or in the network activations they induce (Grosse et al., 2017; Feinman et al., 2017; Xu et al., 2017; Metzen et al., 2017; Carlini & Wagner, 2017a). In this spirit, we propose a method that measures how feature representations and log-odds change under noise: If the input is adversarially perturbed, the noise-induced feature variation tends to have a characteristic direction, whereas it tends not to have any specific direction if the input is natural. We evaluate our method against strong iterative attacks and show that even an adversary aware of the defense cannot evade our detector. E.g. for an L^∞-PGD white-box attack on CIFAR10, our method achieves a detection rate of 99% (FPR $< 1\%$), with accuracies of 96% on clean and 92% on adversarial samples respectively. On ImageNet, we achieve a detection rate of 99% (FPR 1%). Our code can be found at `https://github.com/yk/icml19_public`.

In summary, we make the following contributions:

- We propose a statistical test for the detection and classification of adversarial examples.
- We establish a link between adversarial perturbations and inverse problems, providing valuable insights into the feature space kinematics of adversarial attacks.
- We conduct extensive performance evaluations as well as a range of experiments to shed light on aspects of adversarial perturbations that make them detectable.

[*]Equal contribution [1]Department of Computer Science, ETH Zürich. Correspondence to: <kevin.roth@inf.ethz.ch>, <yannic.kilcher@inf.ethz.ch>, <thomas.hofmann@inf.ethz.ch>.

Proceedings of the 36^{th} International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019.

2. Related Work

Iterative adversarial attacks. Adversarial perturbations are small specifically crafted perturbations of the input, typically imperceptible to humans, that are sufficient to induce large changes in the model output. Let f be a probabilistic classifier with logits f_y and let $F(x) = \arg\max_y f_y(x)$. The goal of the adversary is to find an L^p-norm bounded perturbation $\triangle x \in \mathcal{B}^p_\epsilon(0) := \{\triangle : ||\triangle||_p \leqslant \epsilon\}$, where ϵ controls the attack strength, such that the perturbed sample $x + \triangle x$ gets misclassified by the classifier $F(x)$. Two of the most iconic iterative adversarial attacks are:

Projected Gradient Descent (Madry et al., 2017) aka Basic Iterative Method (Kurakin et al., 2016):

$$
\begin{aligned}
x^0 &\sim \mathcal{U}(\mathcal{B}^p_\epsilon(x)) && (1)\\
x^{t+1} &= \Pi_{\mathcal{B}^\infty_\epsilon(x)}\left(x^t - \alpha\,\mathrm{sign}(\nabla_x \mathcal{L}(f;x,y)|_{x^t})\right) && [L^\infty]\\
x^{t+1} &= \Pi_{\mathcal{B}^2_\epsilon(x)}\left(x^t - \alpha\frac{\nabla_x \mathcal{L}(f;x,y)|_{x^t}}{||\,\nabla_x \mathcal{L}(f;x,y)|_{x^t}\,||_2}\right) && [L^2]
\end{aligned}
$$

where the second and third line refer to the L^∞- and L^2-norm variants respectively, $\Pi_{\mathcal{S}}$ is the projection operator onto the set $\mathcal{S}$, α is a small step-size, y is the target label and $\mathcal{L}(f;x,y)$ is a suitable loss function. For untargeted attacks $y = F(x)$ and the sign in front of α is flipped, so as to ascend the loss function.

Carlini-Wagner attack (Carlini & Wagner, 2017b):

$$
\begin{aligned}
&\text{minimize} \quad ||\triangle x||_p + c\mathcal{F}(x + \triangle x)\\
&\text{such that} \quad x + \triangle x \in \mathrm{dom}_x
\end{aligned}
\qquad (2)
$$

where $\mathcal{F}$ is an objective function, defined such that $\mathcal{F}(x + \triangle x) \leqslant 0$ if and only if $F(x + \triangle x) = y$, e.g. $\mathcal{F}(x) = \max(\max\{f_z(x) : z \neq y\} - f_y(x), -\kappa)$ (see Section V.A in (Carlini & Wagner, 2017b) for a list of objective functions with this property) and dom_x denotes the data domain, e.g. $\mathrm{dom}_x = [0,1]^D$. The constant c trades off perturbation magnitude (proximity) with perturbation strength (attack success rate) and is chosen via binary search.

Detection. The approaches most related to our work are those that defend a machine learning model against adversarial attacks by detecting whether or not the input has been perturbed, either by detecting characteristic regularities in the adversarial perturbations themselves or in the network activations they induce (Grosse et al., 2017; Feinman et al., 2017; Xu et al., 2017; Metzen et al., 2017; Song et al., 2017; Li & Li, 2017; Lu et al., 2017; Carlini & Wagner, 2017a).

Notably, Grosse et al. (2017) argue that adversarial examples are not drawn from the same distribution as the natural data and can thus be detected using statistical tests. Metzen et al. (2017) propose to augment the deep classifier net with a binary "detector" subnetwork that gets input from intermediate feature representations and is trained to discriminate between natural and adversarial network activations. Feinman et al. (2017) suggest to detect adversarial examples by testing whether inputs lie in low-confidence regions of the model either via kernel density estimates in the feature space of the last hidden layer or via dropout uncertainty estimates of the classfier's predictions. Xu et al. (2017) propose to detect adversarial examples by comparing the model's predictions on a given input with its predictions on a squeezed version of the input, such that if the difference between the two exceeds a certain threshold, the input is considered to be adversarial. A quantitative comparison with the last two methods can be found in the Experiments Section.

Origin. It is still an open question whether adversarial examples exist because of intrinsic flaws of the model or learning objective or whether they are solely the consequence of non-zero generalization error and high-dimensional statistics (Gilmer et al., 2018; Schmidt et al., 2018; Fawzi et al., 2018). We note that our method works regardless of the origin of adversarial examples: as long as they induce characteristic regularities in the feature representations of a neural net, e.g. under noise, they can be detected.

3. Identifying and Correcting Manipulations

3.1. Perturbed Log-Odds

We work in a multiclass setting, where pairs of inputs $x^* \in \Re^D$ and class labels $y^* \in \{1, \ldots, K\}$ are generated from a data distribution $\mathbb{P}$. The input may be subjected to an adversarial perturbation $x = x^* + \triangle x$ such that $F(x) \neq y^* = F(x^*)$, forcing a misclassification. A well-known defense strategy against such manipulations is to voluntarily corrupt inputs by noise before processing them. The rationale is that by adding noise $\eta \sim \mathbb{N}$, one may be able to recover the original class, if $\Pr\{F(x + \eta) = y^*\}$ is sufficiently large. For this to succeed, one typically utilizes domain knowledge in order to construct meaningful families of random transformations, as has been demonstrated, for instance, in (Xie et al., 2017; Athalye & Sutskever, 2017). Unstructured (e.g. white) noise, on the other hand, does typically not yield practically viable tradeoffs between probability of recovery and overall accuracy loss.

We thus propose to look for more subtle statistics that can be uncovered by using noise as a *probing instrument* and not as a direct means of recovery. We will focus on probabilistic classifiers with a logit layer of scores as this gives us access to continuous values. For concreteness we will explicitly parameterize logits via $f_y(x) = \langle w_y, \phi(x)\rangle$ with class-specific weight vectors w_y on top of a feature map ϕ realized by a (trained) deep network. Note that typically $F(x) = \arg\max_y f_y(x)$. We also define pairwise log-odds between classes y and z, given input x

$$
f_{y,z}(x) = f_z(x) - f_y(x) = \langle w_z - w_y, \phi(x)\rangle\,. \qquad (3)
$$

We are interested in the noise-perturbed log-odds $f_{y,z}(x+\eta)$ with $\eta \sim \mathbb{N}$, where $y = y^*$, if ground truth is available, e.g. during training, or $y = F(x)$, during testing.

Note that the log-odds may behave differently for different class pairs, as they reflect class confusion probabilities that are task-specific and that cannot be anticipated *a priori*. This can be addressed by performing a Z-score standardization across data points x and perturbations η. For each fixed class pair (y, z) define:

$$\begin{aligned} g_{y,z}(x,\eta) &:= f_{y,z}(x+\eta) - f_{y,z}(x) \\ \mu_{y*,z} &:= \mathbf{E}_{x*|y*}\mathbf{E}_{\eta}\left[g_{y*,z}(x^*,\eta)\right] \\ \sigma^2_{y*,z} &:= \mathbf{E}_{x*|y*}\mathbf{E}_{\eta}\left[(g_{y*,z}(x^*,\eta) - \mu_{y*,z})^2\right] \\ \bar{g}_{y,z}(x,\eta) &:= \left[g_{y,z}(x,\eta) - \mu_{y,z}\right]/\sigma_{y,z}\,. \end{aligned} \tag{4}$$

In practice, all of the above expectations are computed by sample averages over training data and noise instantiations. Also, note that $g_{y,z}(x,\eta) = \langle w_z - w_y, \phi(x+\eta) - \phi(x)\rangle$, i.e. our statistic measures noise-induced feature map weight-difference vector alignment, cf. Section 4.

3.2. Log-Odds Robustness

The main idea pursued in this paper is that the robustness properties of the perturbed log-odds statistics are different, dependent on whether $x = x^*$ is naturally generated or whether it is obtained through an (unobserved) adversarial manipulation, $x = x^* + \triangle x$.

Firstly, note that it is indeed very common to use (small-amplitude) noise during training as a way to robustify models or to use regularization techniques which improve model generalization. In our notation this means that for $(x^*, y^*) \sim \mathbb{P}$, it is a general design goal – prior to even considering adversarial examples – that with high probability $f_{y^*,z}(x^* + \eta) \approx f_{y^*,z}(x^*)$, i.e. that log-odds with regard to the true class remain stable under noise. We generally may expect $f_{y^*,z}(x^*)$ to be negative (favoring the correct class) and slightly increasing under noise, as the classifier may become less certain.

Secondly, we posit that for many existing deep learning architectures, common adversarial attacks find perturbations $\triangle x$ that are *not* robust, but that overfit to specifics of x. We elaborate on this conjecture below by providing empirical evidence and theoretical insights. For the time being, note that *if* this conjecture can be reasonably assumed, then this opens up ways to design statistical tests to identify adversarial examples and even to infer the true class label, which is particularly useful for test time attacks.

Consider the case of a test time attack, where we suspect an unknown perturbation $\triangle x$ has been applied such that $F(x^* + \triangle x) = y \neq y^*$. If the perturbation is not robust w.r.t. the noise process, then this will yield

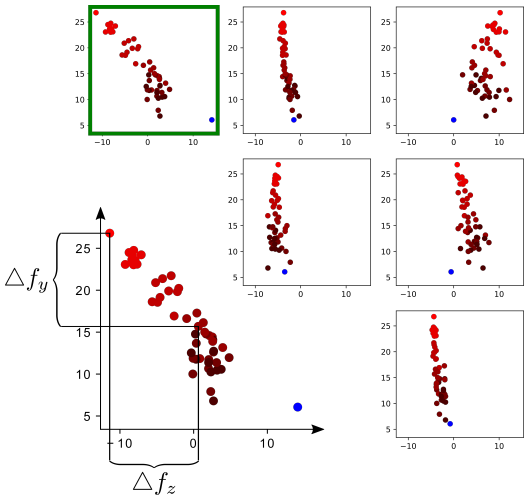

Figure 1. Change of logit scores f_y (on the vertical axis) and f_z (on the horizontal axis) when adding noise to an adversarially perturbed example $x = x^* + \triangle x$. Light red dot: x. Other red dots: $x + \eta$, with color coding of noise amplitude (light = small, dark = large). Light blue dot: x^*. Different plots correspond to different candidate classes z. The candidate class in the green box is selected by Equation 6 and the plot magnified in the lower left.

$f_{y,y*}(x+\eta) > f_{y,y*}(x)$, meaning that noise will partially undo the effect of the adversarial manipulation and directionally revert the log-odds towards the true class y^* in a way that is statistically captured in the perturbed log-odds. Figure 1 (lower left corner) shows this reversion effect. Figure 2 shows an experiment performed on the CIFAR10 data set, which confirms that the histograms of standardized log-odds $\bar{g}_{y,z}(x)$ (defined below) show a good spearation between clean data x^* and manipulated data points $x = x^* + \triangle x$.

3.3. Statistical Test & Corrected Classification

We propose to use the expected perturbed log-odds $\bar{g}_{y,z}(x) = \mathbf{E}_{\eta}\left[\bar{g}_{y,z}(x,\eta)\right]$ as statistics to test whether x classified as y should be thought of as a manipulated example of (true) class z or not. To that extent, we define thresholds $\tau_{y,z}$, which guarantee a maximal false detection rate (of say 1%), yet maximize the true positive rate of identifying adversarial examples. We then flag an example $(x, y := F(x))$ as (possibly) manipulated, if

$$\max_{z \neq y} \{\bar{g}_{y,z}(x) - \tau_{y,z}\} \geqslant 0\,, \tag{5}$$

otherwise it is considered clean.

For test time attacks, it may be relevant not only to detect manipulations, but also to *correct* them on the spot. The simplest approach is to define a new classifier G via

$$G(x) = \arg\max_{z} \{\bar{g}_{y,z}(x) - \tau_{y,z}\}\,, \quad y := F(x)\,. \tag{6}$$

Here we have set $\tau_{y,y} = \bar{g}_{y,y} = 0$, which sets the correct reference point consistent with Equation 5.

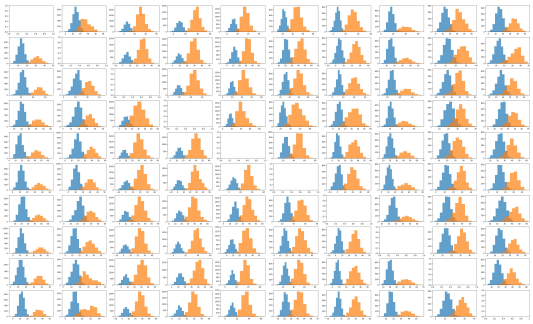

Figure 2. Histograms of the test statistic $\bar{g}_{y,z}(x)$ aggregated over all data points in the training set. Blue represents natural data, orange represents adversarially perturbed data. Columns correspond to predicted labels y, rows to candidate classes z.

A more sophisticated approach is to build a second level classifier on top of the perturbed log-odds statistics. We performed experiments with training a logistic regression classifier for each class y on top of the standardized log-odds scores $\bar{g}_{y,z}(x)$, $y = F(x)$, $z \neq y$. We found this to further improve classification accuracy, especially in cases where several Z-scores are comparably far above the threshold. See Section 7.1 in the Appendix for further details.

4. Feature Space Analysis

4.1. Optimal Feature Space Manipulation

The feature space view allows us to characterize the optimal direction of manipulation for an attack targetting some class z. Obviously the log-odds $f_{y*,z}$ only depend on a single direction in feature space, namely $\triangle w_z = w_z - w_{y*}$.

Proposition 1. *For constraint sets $\mathcal{B}$ that are closed under orthogonal projections, the optimal attack in feature space takes the form $\triangle\phi^* = \alpha(w_z - w_{y*})$ for some $\alpha \geqslant 0$.*

Proof. Assume $\triangle\phi \in \mathcal{B}$ is optimal. We can decompose $\triangle\phi = \alpha\triangle w_z + v$, where $v \perp \triangle w_z$. $\triangle\phi^*$ achieves the same change in log-odds as $\triangle\phi$ and is also optimal. □

Proposition 2. *If $\triangle\phi$ s.t. $y = \arg\max_z \langle w_z, \phi(x) + \triangle\phi\rangle$ and $y^* = \arg\max_z \langle w_z, \phi(x)\rangle$, then $\langle\triangle\phi, \triangle w_y\rangle \geqslant 0$.*

Proof. Follows directly from ϕ-linearity of log-odds. □

Now, as we treat the deep neural net defining ϕ as a black box device, it is difficult to state whether a (near-)optimal feature space attack can be carried out by manipulating in the input space via $\triangle x \mapsto \triangle\phi$. However, we will use some DNN phenomenology as a starting point for making reasonable assumptions that can advance our understanding.

4.2. Pre-Image Problems

The feature space view suggests to search for a pre-image of the optimal manipulation $\phi(x) + \triangle\phi^*$ or at least a manipulation $\triangle x$ such that $\|\phi(x) + \triangle\phi^* - \phi(x + \triangle x)\|^2$ is small. A naïve approach would be to linearize ϕ at x and use the Jacobian,

$$\phi(x + \triangle x) = \phi(x) + J_\phi(x)\triangle x + O(\|\triangle x\|^2)\,. \qquad (7)$$

Iterative improvements could then be obtained by inverting (or pseudo-inverting) $J_\phi(x)$, but are known to be plagued by instabilities. A popular alternative is the so-called Jacobian transpose method from inverse kinematics (Buss, 2004; Wolovich & Elliott, 1984; Balestrino et al., 1984). This can be motivated by a simple observation

Proposition 3. *Given an input x as well as a target direction $\triangle\phi$ in feature space. Define $\triangle x := J_\phi^\top(x)\triangle\phi$ and assume that $\langle J_\phi\triangle x, \triangle\phi\rangle > 0$. Then there exists an $\epsilon > 0$ (small enough) such that $x^+ := x + \epsilon\triangle x$ is a better pre-image in that $\|\phi(x) + \triangle\phi - \phi(x^+)\| < \|\triangle\phi\|$.*

Proof. Follows from Taylor expansion of ϕ. □

It turns out that by the chain rule, we get for any loss ℓ defined in terms of features ϕ,

$$\nabla_x(\ell \circ \phi)(x) = J_\phi^\top(x)\nabla_\phi\ell(\phi)|_{\phi=\phi(x)}\,. \qquad (8)$$

With the soft-max loss $\ell(x) = -f_y(x) + \log\sum_z \exp[f_z(x)]$ and in case of $f_{y*}(x) \gg f_z(x)$ one gets

$$\nabla_\phi\ell(\phi) = w_{y*} - w_y = -\triangle w_y\,. \qquad (9)$$

This shows that a gradient-based iterative attack is closely related to solving the pre-image problem for finding an optimal feature perturbation via the Jacobian transpose method.

4.3. Approximate Rays and Adversarial Cones

If an adversary had direct control over the feature space representation, optimal attack vectors could always be found along the ray $\triangle w_z$. As the adversary has to work in input space, this may only be possible in approximation however. Experimentally, we have found that an optimal perturbation typically defines a ray in input space, $x + t\triangle x$ ($t \geqslant 0$), yielding a feature-space trajectory $\phi(t) = \phi(x + t\triangle x)$ for which the rate of change along $\triangle w_z$ is nearly constant over a relevant range of t, see Figures 3 & 9. While the existence of such rays obviously plays in the hand of an adversary, it remains an open theoretical question to eluciate properties of the model architecture causing such vulnerabilities.

As adversarial directions are expected to be suscpetible to angular variations (otherwise they would be simple to find and pointing at a general lack of model *robustness*), we conjecture that geometrically optimal adversarial manipulations are embedded in a cone-like structure, which we call *adversarial cone*. Experimental evidence for the existence of such cones is visualized in Figure 5. It is a virtue of the commutativity of applying the adversarial $\triangle x$ and random noise η that our statistical test can reliably detect such adversarial cones.

Table 1. Baseline test set accuracies on clean and PGD-perturbed examples for the models we considered.

DATASET	MODEL	TEST SET ACCURACY (CLEAN / PGD)
CIFAR10	WRESNET	96.2% / 2.60%
	CNN7	93.8% / 3.91%
	CNN4	73.5% / 14.5%
IMAGENET	INCEPTION V3	76.5% / 7.2%
	RESNET 101	77.0% / 7.2%
	RESNET 18	69.2% / 6.5%
	VGG11(+BN)	70.1% / 5.7%
	VGG16(+BN)	73.3% / 6.1%

5. Experimental Results

5.1. Datasets, Architectures & Training Methods

In this section, we provide experimental support for our theoretical propositions and we benchmark our detection and correction methods on various architectures of deep neural networks trained on the CIFAR10 and ImageNet datasets. For CIFAR10, we compare the WideResNet implementation of Madry et al. (2017), a 7-layer CNN with batch normalization and a vanilla 4-layer CNN. In the following, if nothing else is specified, we use the 7-layer CNN as a default platform, since it has good test set accuracy at relatively low computational requirements. For ImageNet, we use a selection of models from the torchvision package (Marcel & Rodriguez, 2010), including Inception V3, ResNet101 and VGG16. Further details can be found in the Appendix.

As a default attack strategy we use an L^∞-norm constrained PGD white-box attack. The attack budget ϵ_∞ was chosen to be the smallest value such that almost all examples are successfully attacked. For CIFAR10 this is $\epsilon_\infty = 8/255$, for ImageNet $\epsilon_\infty = 2/255$. We experimented with a number of different PGD iterations and found that the detection rate and corrected classification accuracy are nearly constant across the entire range from 10 up to 1000 attack iterations, as shown in Figure 8 in the Appendix. For the remainder of this paper, we thus fixed the number of attack iterations to 20. Table 1 shows test set accuracies for all considered models on both clean and adversarial samples.

We note that the detection test in Equation 5 as well as the basic correction algorithm in Equation 6 are completely attack agnostic. The only stage that explicitly includes an adversarial attack model is the second-level logistic classifier based correction algorithm, which is trained on adversarially perturbed samples, as explained in Section 7.1 in the Appendix. While this could in principle lead to overfitting to the particular attacks considered, we empirically show that the second-level classifier based correction algorithm performs well under attacks not seen during training, cf. Section 5.6, as well as specifically designed counter-attacks, cf. Section 5.7.

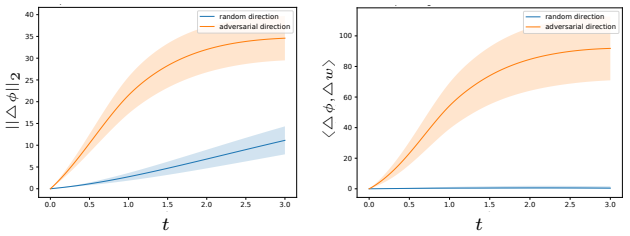

Figure 3. (Left) Norm of the induced feature space perturbation along adversarial and random directions. (Right) Weight-difference alignment. For the adversarial direction, the alignment with the weight-difference between the true and adversarial class is shown. For the random direction, the largest alignment with any weight-difference vector is shown. See also Figure 8.

5.2. Detectability of Adversarial Examples

Induced feature space perturbations. We compare the norm of the induced feature space perturbation $||\triangle\phi||_2$ along adversarial directions $x^* + t\triangle x$ with that along random directions $x^* + t\eta$ (where the expected norm of the noise is set to be approximately equal to the expected norm of the adversarial perturbation). We also compute the alignment $\langle\triangle\phi, \triangle w\rangle$ between the induced feature space perturbation and certain weight-difference vectors: For the adversarial direction, the alignment is computed w.r.t. the weight-difference vector between the true and adversarial class, for the random direction, the largest alignment with any weight-difference vector is computed.

The results are reported in Figure 3. The plot on the left shows that iterative adversarial attacks induce feature space perturbations that are significantly larger than those induced by random noise. The plot on the right shows that the attack-induced weight-difference alignment is significantly larger than the noise-induced one. The plot on the right in Figure 8 in the Appendix further shows that the noise-induced weight-difference alignment is significantly larger for the adversarial example than for the natural one. Combined, this indicates that *adversarial examples cause atypically large feature space perturbations along the weight-difference direction* $\triangle w_y = w_y - w_{y*}$, *with* $y = F(x)$.

Distance to decision boundary. Next, we investigate how the distance to the decision boundary for adversarial examples compares with that of their unperturbed counterpart. To this end, we measure the logit cross-over when linearly interpolating between an adversarially perturbed example and its natural counterpart, i.e. we measure $t \in [0, 1]$ s.t. $f_{y*}(x^* + t\triangle x) \simeq f_y(x^* + t\triangle x)$, where $y = F(x^* + \triangle x)$. We also measure the average L^2-norm of the DeepFool perturbation $\triangle x(t)$, required to cross the nearest decision boundary[1], for all interpolants $x(t) = x^* + t\triangle x$.

[1]The DeepFool attack aims to find the shortest path to the nearest decision boundary. We additionally augment DeepFool by a binary search to hit the decision boundary precisely.

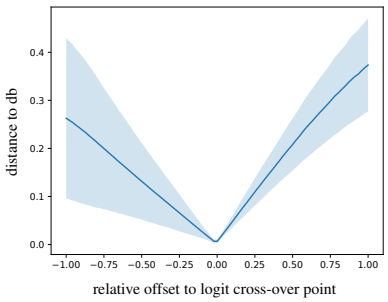

Figure 4. Average distance to the decision boundary when interpolating from natural examples to adversarial examples. The horizontal axis shows the relative offset of the interpolant $x(t)$ to the logit cross-over point located at the origin 0. For each interpolant $x(t)$, the distance to the nearest decision boundary is computed. The plot shows that natural examples are slightly closer to the decision boundary than adversarial examples.

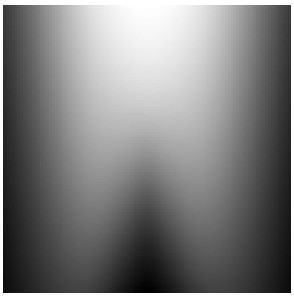

Figure 5. Adversarial cone. The plot shows the averaged softmax prediction for the natural class over ambient space hyperplanes spanned by the adversarial perturbation (on the vertical axis) and randomly sampled orthogonal vectors (on the horizontal axis). The natural sample is located one-third from the top, the adversarial sample one third from the bottom on the vertical axis through the middle of the plot.

We find that the mean logit cross-overs is at $\bar{t} = 0.43$. Similarly, as shown in Figure 4, the mean L^2-distance to the nearest decision boundary is 0.37 for adversarial examples, compared to 0.27 for natural ones. Hence, natural examples are even slightly closer to the decision boundary. *We can thus rule out the possibility that adversarial examples are detectable because of a trivial discrepancy in distance to the decision boundary*.

Neighborhood of adversarial examples. We measure the ratio of the 'distance between the adversarial and the corresponding unperturbed example' to the 'distance between the adversarial example and the nearest other neighbor (in either training or test set)', i.e. we compute $||x-x^*||_2/||x-x^{\text{nn}}||_2$ over a number of samples in the test set, for various L^∞- & L^2-bounded PGD attacks (with $\epsilon_2 = \sqrt{D}\epsilon_\infty$).

We consistently find that the ratio is sharply peaked around a value much smaller than one. E.g. for L^∞-PGD attack with $\epsilon_\infty = 8/255$ we get 0.075 ± 0.018, while for the corresponding L^2-PGD attack we obtain 0.088 ± 0.019. Further values can be found in Table 7 in the Appendix. We note that similar findings have been reported by Tramèr et al. (2017). Hence, "perceptually similar" *adversarial examples are much closer to the unperturbed sample than to any other neighbor in the training or test set*.

We would therefore naturally expect that the feature representation is more likely to be shifted to the original unperturbed class rather than any other neighboring class when the adversarial example is convolved with random noise.

To investigate this further, we plot the softmax predictions when adding noise to the adversarial example. The results are reported in Figure 9 in the Appendix. The plot on the left shows that the probability of the natural class increases faster than the probability of the highest other class when adding noise with a small to intermediate magnitude to the adversarial example. However, the probability of the natural class never climbs to be the highest probability of all classes, which is why simple addition of noise to an adversarial example does not recover the natural class in general.

Adversarial Cones. To visualize the ambient space neighborhood around natural and adversarially perturbed samples, we plot the averaged classifier prediction for the natural class over hyperplanes spanned by the adversarial perturbation and randomly sampled orthogonal vectors, i.e. we plot $\mathbf{E}_n[F_{y*}(x^* + t\triangle x + sn)]$ for $s \in [-1, 1], t \in [-1, 2]$ with s along the horizontal and t along the vertical axis, where F_{y*} denotes the softmax and $\mathbf{E}_n$ denotes expectation over random vectors $n \perp \triangle x$ with approximately equal norm.

Interestingly, the plot reveals that adversarial examples are embedded in a cone-like structure, i.e. the adversarial sample is statistically speaking "surrounded" by the natural class, as can be seen from the gray rays confining the adversarial cone. This confirms our theoretical argument that the noise-induced feature variation tends to have a direction that is indicative of the natural class when the input is adversarially perturbed.

It is a virtue of the commutativity of applying the adversarial and random noise, i.e. $x^* + \triangle x + \eta$ vs. $x^* + \eta + \triangle x$, that our method can reliably detect such adversarial cones.

5.3. Detection rates and classification accuracies

In the remainder of this section we present the results of various performance evaluations. The reported detection rates measure how often our method classifies a sample as being adversarial, corresponding to the False Positive Rate if the sample is clean and to the True Positive Rate if it was perturbed. We also report accuracies for the predictions made by the logistic classifier based correction method.

Table 2. Detection rates of our statistical test.

DATASET	MODEL	DETECTION RATE (CLEAN / PGD)
CIFAR10	WRESNET	0.2% / 99.1%
	CNN7	0.8% / 95.0%
	CNN4	1.4% / 93.8%
IMAGENET	INCEPTION V3	1.9% / 99.6%
	RESNET 101	0.8% / 99.8%
	RESNET 18	0.6% / 99.8%
	VGG11(+BN)	0.5% / 99.9%
	VGG16(+BN)	0.3% / 99.9%

Table 3. Accuracies of our correction method.

DATASET	MODEL	ACCURACY (CLEAN / PGD)
CIFAR10	WRESNET	96.0% / 92.7%
	CNN7	93.6% / 89.5%
	CNN4	71.0% / 67.6%

Tables 2 and 3 report the detection rates of our statistical test and accuracies of the corrected predictions. Our method manages to detect nearly all adversarial samples, seemingly getting better as models become more complex, while the false positive rate stays around 1%. Further[2], our second-level logistic-classifier based correction method manages to reclassify almost all of the detected adversarial samples to their respective source class successfully, resulting in test set accuracies on adversarial samples within 5% of the respective test set accuracies on clean samples. Also note that due to the low false positive rate, the drop in performance on clean samples is negligible.

5.4. Effective strength of adversarial perturbations.

We measure how the detection rate and reclassification accuracy of our method depend on the effective attack strength. To this end, we define the effective Bernoulli-q strength of ϵ-bounded adversarial perturbations as the attack success rate when each entry of the perturbation $\triangle x$ is individually accepted with probability q and set to zero with probability $1-q$. For $q=1$ we obtain the usual adversarial misclassification rate. We naturally expect weaker attacks to be less effective but also harder to detect than stronger ones.

The results are reported in Figure 6. We can see that the uncorrected accuracy of the classifier decreases monotonically as the effective attack strength increases, both in terms of the attack budget ϵ_∞ as well as in term of the fraction q of accepted perturbation entries. Meanwhile, the detection rate of our method increases at such a rate that the corrected classifier manages to compensate for the decay in uncorrected accuracy, due to the decrease in effective strength of the perturbations, across the entire range considered.

[2]Due to computational constraints, we focus on the CIFAR10 models in the remainder of this paper.

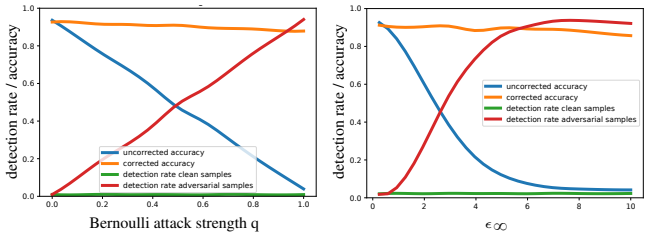

Figure 6. Detection rate and reclassification accuracy as a function of the effective attack strength. The uncorrected classifier accuracy decreases as the attack strength increases, both in terms of the attack budget ϵ_∞ as well as in terms of the fraction q of accepted perturbation entries. Meanwhile, the detection rate of our method increases at such a rate that the corrected classifier manages to compensate for the decay in uncorrected accuracy.

Table 4. Test set accuracies for adversarially trained models.

DATASET	ADVERSARIALLY TRAINED MODEL	ACCURACY (CLEAN / PGD)
CIFAR10	WRESNET	87.3% / 55.2%
	CNN7	82.2% / 44.4%
	CNN4	68.2% / 40.4%

5.5. Comparison to Adversarial Training

For comparison, we also report test set and white-box attack accuracies for adversarially trained models. Madry et al. (2017)'s WResNet was available as an adversarially pretrained variant, while the other models were adversarially trained as outlined in Section 7.1 in the Appendix. The results for the best performing classifiers are shown in Table 4. We can see that the accuracy on adversarial samples is significantly lower while the drop in performance on clean samples is considerably larger for adversarially trained models compared to our method.

5.6. Defending against unseen attacks

Next, we evaluate our method on adversarial examples created by attacks that are different from the L_∞-constrained PGD attack used to train the second-level logistic classifier. The rationale is that the log-odds statistics of the unseen attacks could be different from the ones used to train the logistic classifier. We thus want to test whether it is possible to evade correct reclassification by switching to a different attack. As alternative attacks we use an L^2-constrained PGD attack as well as the L^2-Carlini-Wagner attack.

The baseline accuracy of the undefended CNN7 on adversarial examples is 4.8% for the L^2-PGD attack and 3.9% for the Carlini-Wagner attack. Table 5 shows the detection rates and corrected classification accuracies of our method. As can be seen, there is only a slight decrease in performance, i.e. our method remains capable of detecting and correcting most adversarial examples of the previously unseen attacks.

Table 5. CIFAR10 detection rates and reclassification accuracies on adversarial samples from attacks that have not been used to train the second-level logistic classifier.

ATTACK	DETECTION RATE (CLEAN / ATTACK)	ACCURACY (CLEAN / ATTACK)
L^2-PGD	1.0% / 96.1%	93.3% / 92.9%
L^2- CW	4.8% / 91.6%	89.7% / 77.9%

Table 6. CIFAR10 detection rates and reclassification accuracies on clean and adversarial samples from the defense-aware attacker.

MODEL	DETECTION RATE (CLEAN / ATTACK)	ACCURACY (CLEAN / ATTACK)
WRESNET	4.5% / 71.4%	91.7% / 56.0%
CNN7	2.8% / 75.5%	91.2% / 56.6%
CNN4	4.1% / 81.3%	69.0% / 56.5%

5.7. Defending against defense-aware attacks

Finally, we evaluate our method in a setting where the attacker is fully aware of the defense, in order to see if the defended network is susceptible to cleverly designed counter-attacks. Since our defense is built on random sampling from noise sources that are under our control, the attacker will want to craft perturbations that perform well *in expectation* under this noise. The optimality of this strategy in the face of randomization-based defenses was established in Carlini & Wagner (2017a) (cf. their recipe to attack the dropout randomization defense of Feinman et al. (2017)). Specifically, each PGD perturbation is computed for a loss function that is an empirical average over $K = 100$ noise-convolved data points, with the same noise source as used for detection. (We have also experimented with other variants such as backpropagating through our statistical test and found the above approach by Carlini & Wagner (2017a) to work best.)

The undefended accuracies under this attack for the models under consideration are: WResNet 2.8%, CNN7 3.6% and CNN4 14.5%. Table 6 shows the corresponding detection rates and accuracies after defending with our method. Compared to Section 5.6, the drop in performance is larger, as we would expect for a defense-aware counter-attack, however, both the detection rates and the corrected accuracies remain remarkably high compared to the undefended network.

5.8. Comparison with related detection methods

In this last section we provide a quantitative comparison with two of the leading detection methods: feature squeezing of Xu et al. (2017) and dropout randomization (aka Bayesian neural network uncertainty) of Feinman et al. (2017). The reason we compare against those two is that Carlini & Wagner (2017a) consider dropout randomization to be the only defense, among the ten methods they surveyed (including the other two detection methods we mentioned in more detail in the related work section), that is not completely broken, while the more recent feature squeezing method was selected because it was evaluated extensively on comparable settings to ours.

On CIFAR10, feature squeezing[3] (DenseNet) significantly enhances the model robustness against L^2-CW attacks, while it is considerably less effective against PGD attacks, which the authors suspect could due to feature squeezing being better suited to mitigating smaller perturbations. For L^2-CW attacks, they report a detection rate of 100% (FPR $< 5\%$) and corrected accuracies of 89% on clean and 83% on adversarial examples, which is slightly better than our numbers in Table 5. We would like to note however that we calibrated our method on L^∞-constrained perturbations and that our numbers could probably be improved by calibrating on L^2-constrained perturbations instead. For L^∞-PGD, feature squeezing achieves a detection rate of 55% (FPR $< 5\%$), with corrected accuracies of 89% on clean and 56% on adversarial examples, whereas our method achieves a detection rate of 99% (FPR $< 1\%$), with accuracies of 96% on clean and 92% on adversarial samples respectively. On ImageNet, feature squeezing (MobileNet) achieves a detection rate of 64% (FPR 5%) for L^∞-PGD, while our method achieves a detection rate of 99% (FPR 1%). Xu et al. (2017) only evaluate feature squeezing against defense-aware attacks on MNIST, finding that their method is not immune.

Feinman et al. (2017) do not report individual true and false detection rates. They do however show the ROC curve and report its AUC: compare their BIM-B curve (PGD with fixed number of iterations) in Figures 9c & 10 with our Figure 10 in the Appendix. On CIFAR10 (ResNet) Carlini & Wagner (2017a) were able to fool the dropout defense with 98% success, i.e. the detection rate is 2% for the defense-aware L^2-CW attack. Our method achieves a detection rate of 71.4% (FPR 4.5%) in a comparable setting.

6. Conclusion

We have shown that adversarial examples exist in cone-like regions in very specific directions from their corresponding natural examples. Based on this, we design a statistical test of a given sample's log-odds robustness to noise that can predict with high accuracy if the sample is natural or adversarial and recover its original class label, if necessary. Further research into the properties of network architectures is necessary to explain the underlying cause of this phenomenon. It remains an open question which current model families follow this paradigm and whether criteria exist which can certify that a given model is immunizable via our method.

[3] We report their best joint detection ensemble of squeezers.

Acknowledgements

We would like to thank Sebastian Nowozin, Aurelien Lucchi, Michael Tschannen, Gary Becigneul, Jonas Kohler and the dalab team for insightful discussions and helpful comments.

References

Athalye, A. and Sutskever, I. Synthesizing robust adversarial examples. *arXiv preprint arXiv:1707.07397*, 2017.

Athalye, A., Carlini, N., and Wagner, D. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. *arXiv preprint arXiv:1802.00420*, 2018.

Balestrino, A., De Maria, G., and Sciavicco, L. Robust control of robotic manipulators. *IFAC Proceedings Volumes*, 17(2):2435–2440, 1984.

Biggio, B., Corona, I., Maiorca, D., Nelson, B., Srndic, N., Laskov, P., Giacinto, G., and Roli, F. Evasion attacks against machine learning at test time. In *Joint European conference on machine learning and knowledge discovery in databases*, pp. 387–402. Springer, 2013.

Buss, S. R. Introduction to inverse kinematics with jacobian transpose, pseudoinverse and damped least squares methods. *IEEE Journal of Robotics and Automation*, 17 (1-19):16, 2004.

Carlini, N. and Wagner, D. Adversarial examples are not easily detected: Bypassing ten detection methods. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pp. 3–14. ACM, 2017a.

Carlini, N. and Wagner, D. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy (SP)*, pp. 39–57. IEEE, 2017b.

Fawzi, A., Fawzi, H., and Fawzi, O. Adversarial vulnerability for any classifier. *arXiv preprint arXiv:1802.08686*, 2018.

Feinman, R., Curtin, R. R., Shintre, S., and Gardner, A. B. Detecting adversarial samples from artifacts. *arXiv preprint arXiv:1703.00410*, 2017.

Gilmer, J., Metz, L., Faghri, F., Schoenholz, S. S., Raghu, M., Wattenberg, M., and Goodfellow, I. Adversarial spheres. *arXiv preprint arXiv:1801.02774*, 2018.

Goodfellow, I. J., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

Grosse, K., Manoharan, P., Papernot, N., Backes, M., and McDaniel, P. On the (statistical) detection of adversarial examples. *arXiv preprint arXiv:1702.06280*, 2017.

Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A.-r., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T. N., et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97, 2012.

Kilcher, Y. and Hofmann, T. The best defense is a good offense: Countering black box attacks by predicting slightly wrong labels. *arXiv preprint arXiv:1711.05475*, 2017.

Kurakin, A., Goodfellow, I., and Bengio, S. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533*, 2016.

LeCun, Y., Bengio, Y., and Hinton, G. Deep learning. *nature*, 521(7553):436, 2015.

Li, X. and Li, F. Adversarial examples detection in deep networks with convolutional filter statistics. In *ICCV*, pp. 5775–5783, 2017.

Lu, J., Issaranon, T., and Forsyth, D. A. Safetynet: Detecting and rejecting adversarial examples robustly. In *ICCV*, pp. 446–454, 2017.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

Marcel, S. and Rodriguez, Y. Torchvision the machine-vision package of torch. In *Proceedings of the 18th ACM international conference on Multimedia*, pp. 1485–1488. ACM, 2010.

Metzen, J. H., Genewein, T., Fischer, V., and Bischoff, B. On detecting adversarial perturbations. *arXiv preprint arXiv:1702.04267*, 2017.

Moosavi Dezfooli, S. M., Fawzi, A., and Frossard, P. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, number EPFL-CONF-218057, 2016.

Papernot, N., McDaniel, P., and Goodfellow, I. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *arXiv preprint arXiv:1605.07277*, 2016.

Sabour, S., Cao, Y., Faghri, F., and Fleet, D. J. Adversarial manipulation of deep representations. *arXiv preprint arXiv:1511.05122*, 2015.

Schmidt, L., Santurkar, S., Tsipras, D., Talwar, K., and Madry, A. Adversarially robust generalization requires more data. *arXiv preprint arXiv:1804.11285*, 2018.

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations (ICLR)*, 2014.

Song, Y., Kim, T., Nowozin, S., Ermon, S., and Kushman, N. Pixeldefend: Leveraging generative models to understand and defend against adversarial examples. *arXiv preprint arXiv:1710.10766*, 2017.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

Tramèr, F., Papernot, N., Goodfellow, I., Boneh, D., and McDaniel, P. The space of transferable adversarial examples. *arXiv preprint arXiv:1704.03453*, 2017.

Wolovich, W. A. and Elliott, H. A computational technique for inverse kinematics. In *Decision and Control, 1984. The 23rd IEEE Conference on*, volume 23, pp. 1359–1363. IEEE, 1984.

Xie, C., Wang, J., Zhang, Z., Ren, Z., and Yuille, A. Mitigating adversarial effects through randomization. *arXiv preprint arXiv:1711.01991*, 2017.

Xu, W., Evans, D., and Qi, Y. Feature squeezing: Detecting adversarial examples in deep neural networks. *arXiv preprint arXiv:1704.01155*, 2017.