Adversarial Examples Are Not Easily Detected: Bypassing Ten Detection Methods

Nicholas Carlini David Wagner
University of California, Berkeley

ABSTRACT

Neural networks are known to be vulnerable to adversarial examples: inputs that are close to natural inputs but classified incorrectly. In order to better understand the space of adversarial examples, we survey ten recent proposals that are designed for *detection* and compare their efficacy. We show that *all* can be defeated by constructing new loss functions. We conclude that adversarial examples are significantly harder to detect than previously appreciated, and the properties believed to be intrinsic to adversarial examples are in fact not. Finally, we propose several simple guidelines for evaluating future proposed defenses.

1 INTRODUCTION

Recent years have seen rapid growth in the area of machine learning. Neural networks, an idea that dates back decades, have been a driving force behind this rapid advancement. Their successes have been demonstrated in a wide set of domains, from classifying images [38], to beating humans at Go [35], to NLP [32, 40], to self driving cars [6].

In this paper, we study neural networks applied to image classification. While neural networks are the most accurate machine learning approach known to date, they are against an adversary who attempts to fool the classifier [5]. That is, given a natural image x, an adversary can easily produce a visually similar image x' that has a different classification. Such an instance x' is known as an *adversarial example* [39], and they have been shown to exist in nearly all domains that neural networks are used.

The research community has reacted to this observation in force, proposing many defenses that attempt to classify adversarial examples correctly [3, 16, 20, 21, 31, 33, 34, 41]. Unfortunately, most of these defenses are not effective at classifying adversarial examples correctly.

Due to this difficulty, recent work has turned to attempting to *detect* them instead. We study ten detection schemes proposed in seven papers over the last year [4, 11, 12, 15, 18, 19, 24], and compare their efficacy with the other defenses in a consistent manner. With new attacks, we show that in every case the defense can be evaded by an adversary who targets that specific defense. On simple datasets, the attacks slightly increase the distortion required, but on more complex datasets, adversarial examples remain completely indistinguishable from the original images.

By studying these recent schemes that detect adversarial examples, we challenge the assumption that adversarial examples have intrinsic differences from natural images. We also use these experiments to obtain a better understanding of the space of adversarial examples.

We evaluate these defenses under three threat models. We first consider a generic attacks that don't take any specific measures to fool any particular detector. We show that six of the ten defenses are significantly less effective than believed (although not completely broken) under this threat model. Second, we introduce novel white-box attacks that break each defense when tailored to the given defense; five of the defenses provide *no* increase in robustness, and a further three increase robustness only slightly, and two are marginally effective only on simple datasets. At a technical level, our attacks work by defining a special attacker-loss function that captures the requirement that the adversarial examples must fool the defense, and optimizing for this loss function. We discover that the specific loss function chosen is critical to effectively defeating the defense: choosing the immediately obvious loss function often results in the defense appearing significantly more robust than it actually is. Finally, we show that our white-box attacks can leverage the transferability [39] property to work even when the adversary does not have knowledge of the defense's model parameters.

Our results further suggest that there is a need for better ways to evaluate potential defenses. We believe our approach would be a useful baseline: to be worth considering, a proposed defense should follow the approach used here as a first step towards arguing robustness.

The code to reproduce our results is available online at http://nicholas.carlini.com/code/nn_breaking_detection.

We make the following contributions:

- We find that many defenses are unable to detect adversarial examples, even when the attacker is oblivious to the specific defense used.
- We break all existing detection methods in the white-box (and black-box) setting by showing how to pick good attacker-loss functions for each defense.
- We draw conclusions about the space of adversarial examples, and offer a note of caution about evaluating solely on MNIST; it appears that MNIST has somewhat different security properties than, for instance, CIFAR.
- We provide recommendations for evaluating defenses.

2 BACKGROUND

The remainder of this section contains a brief survey of the field of neural networks and adversarial machine learning. We encourage

AISec'17, November 3, 2017, Dallas, TX, USA

ACM ISBN 978-1-4503-5202-4/17/11…$15.00
https://doi.org/10.1145/3128572.3140444

readers unfamiliar with this area to read the following papers (in this order): [39], [13], [29], and [8].

2.1 Notation

Let $F(\cdot)$ denote a neural network used for classification. The final layer in this network is a softmax activation, so that the output is a probability distribution where $F(x)_i$ represents the probability that object x is labeled with class i.

All neural networks we study are feed-forward networks consisting of multiple layers F^i taking as input the result of previous layers. The outputs of the final layer are known as logits; we represent them by $Z(\cdot)$. Some layers involve the non-linear ReLU [27] activation. Thus the ith layer computes

$$F^i(x) = \text{ReLU}(A^i \cdot F^{i-1}(x) + b^i)$$

where A^i is a matrix and b^i is a vector. Let $Z(x)$ denote the output of the last layer (before the softmax), i.e., $Z(x) = F^n(x)$. Then the final output of the network is

$$F(x) = \text{softmax}(Z(x)).$$

When we write $C(x)$ we mean the classification of $F(\cdot)$ on x:

$$C(x) = \arg\max_i(F(x)_i).$$

Along with the neural network, we are given a set of training instances with their corresponding labels $(x, y) \in \mathcal{X}$.

2.2 Adversarial Examples

The security of machine learning is a well studied field: early work considered this problem mostly on linear classifiers [9, 25]; later work more generally examined the security of machine learning [1, 2] to both evasion and poising attacks.

More recently, Biggio *et al.* and Szegedy *et al.* [5, 39] demonstrated test-time evasion attacks on neural networks. They were able to produce visually similar images that had different labels assigned by the classifier.

We begin by defining an input to the classifier $F(\cdot)$ *natural* if it is an instance that was benignly created (e.g., all instances in the training set and testing set are natural instances). Then, given a network $F(\cdot)$ and a natural input x so that $C(x) = l$ we say that x' is an (untargeted) *adversarial example* if x' is close to x and $C(x') \neq l$. A more restrictive case is where the adversary picks a target $t \neq l$ and seeks to find x' close to x such that $C(x') = t$; in this case we call x' a *targeted* adversarial example. We focus on targeted adversarial examples exclusively in this paper. When we say a neural network is *robust* we mean that it is difficult to find adversarial examples on it.

To define closeness, most attacks use an L_p distance, defined as $\|d\|_p = (\sum_{i=0}^{n} |v_i|^p)^{\frac{1}{p}}$. Common choices of p include: L_0, a measure of the number of pixels changed [30]; L_2, the standard Euclidean norm [8, 26, 39]; or L_∞, a measure of the maximum absolute change to any pixel [13]. If the total distortion under any of these three distance metrics is small, the images will likely appear visually similar. We quantitatively measure the robustness of a defense in this paper by measuring the distance to the nearest adversarial example under the L_2 metric.

One further property of adversarial examples we will make use of is the transferability property [13, 39]. It is often the case that, when given two models $F(\cdot)$ and $G(\cdot)$, an adversarial example on F will also be an adversarial example on G, even if they are trained in completely different manners, on completely different training sets.

There has been a significant amount of work studying methods to construct adversarial examples [5, 8, 13, 26, 30, 39] and to make networks robust against adversarial examples [3, 16, 20, 21, 31, 33, 34, 41]. To date, no defenses has been able to classify adversarial examples correctly.

Given this difficulty in correctly classifying adversarial examples, recent defenses have instead turned to detecting adversarial examples and reject them. We study these defenses in this paper [4, 11, 12, 15, 18, 19, 24].

2.3 Threat Model

As done in Biggio *et al.* [5], we consider three different threat models in this paper:

(1) An *Zero-Knowledge Adversary* generates adversarial examples on the unsecured model F and is not aware that the detector D is in place. The detector is successful if it can detect these adversarial examples.
(2) A *Perfect-Knowledge Adversary* is aware the neural network is being secured with a given detection scheme D, knows the model parameters used by D, and can use these to attempt to evade both the original network F and the detector simultaneously.
(3) A *Limited-Knowledge Adversary* is aware the neural network is being secured with a given detection scheme, knows how it was trained, but does not have access to the trained detector D (or the exact training data).

We evaluate each defense under these three threat models. We discuss our evaluation technique in Section 2.7.

2.4 Datasets

In this paper we consider two datasets used throughout the existing work in this field.

The *MNIST* dataset [23] consists of $70,000$ 28×28 greyscale images of handwritten digits from 0 to 9. Our standard convolutional network achieves 99.4% accuracy on this dataset.

The *CIFAR-10* dataset [22] consists of $60,000$ 32×32 color images of ten different objects (e.g., truck, airplane, etc). This dataset is substantially more difficult: the state of the art approaches achieve 95% accuracy [36]. For comparison with prior work, we use the ResNet [17] architecture from Metzen *et al.* [18] trained in the same manner. This model achieves a 91.5% accuracy.

The first row of Figure 1 shows natural examples drawn from the test set of these datasets.

2.5 Defenses

In order to better understand what properties are intrinsic of adversarial examples and what properties are only artificially true because of existing attack techniques, we choose the first seven papers released that construct defenses to detect adversarial examples.

Three of the defenses [12, 15, 18] use a second neural network to classify images as natural or adversarial. Three use PCA to detect

statistical properties of the images or network paramaters [4, 19, 24]. Two perform other statistical tests [11, 15], and the final two perform input-normalization with randomization and blurring [11, 24].

We summarize our results in Figure 1. Some defenses can slightly increase distortion required for MNIST digits. However, no defense makes CIFAR adversarial examples visually distinguishable from the original image. We generate adversarial examples as described below.

2.6 Generating Adversarial Examples

We use the L_2 attack algorithm of Carlini and Wagner [8] to generate targeted adversarial examples, as it is superior to other published attacks. At a high level it is an iterative attack as done in the initial work on constructing adversarial examples [5, 38]. Given a neural network F with logits Z, the attack uses gradient descent to solve

$$\text{minimize} \quad \|x' - x\|_2^2 + c \cdot \ell(x')$$

where the loss function ℓ is defined as

$$\ell(x') = \max(\max\{Z(x')_i : i \neq t\} - Z(x')_t, -\kappa).$$

We now give some intuition behind this loss function. The difference $\max\{Z(x')_i : i \neq t\} - Z(x')_t$ is used to compare the target class t with the next-most-likely class. However, this is minimized when the target class is significantly more likely than the second most likely class, which is not a property we want. This is fixed by taking the maximum of this quantity with $-\kappa$, which controls the confidence of the adversarial examples. When $\kappa = 0$, the adversarial examples are called *low-confidence adversarial examples* and are only just classified as the target class. As κ increases, the model classifies the adversarial example as increasingly more likely, we call these *high-confidence adversarial examples*.

The constant c is chosen via binary search. If c is too small, the distance function dominates and the optimal solution will not have a different label. If c is too large, the objective term dominates and the adversarial example will not be nearby.

Of critical importance is that the loss function operates over the logits Z, and not the probabilities F. As described in [8], the optimal choice of the constant $c \sim \frac{1}{|\nabla \ell|}$; therefore, if F were used instead of Z, no "good" constant c would exist since f varies by several orders of magnitude (and Z usually only by one). When constructing attacks in later sections, we often choose new loss functions ℓ that also do not vary in their magnitude.

Aside from C&W's attack, the *Fast Gradient Sign* attack [13] and *JSMA* [30] are two attacks used by some defenses for evaluation. These attacks are weaker than C&W's attack and we do not use them for evaluation [8].

2.7 Attack Approach

In order to evaluate the robustness of each of the above defenses, we take three approaches to target each of the three threat models introduced earlier.

Evaluate with a strong attack (Zero-Knowledge): In this step we generate adversarial examples with C&W's attack and check whether the defense can detect this strong attack. This evaluation approach has the weakest threat model (the attacker is not even aware the defense is in place), so any defense should trivially be able to detect this attack. Failing this test implies that the second two tests will also fail.

Perform an adaptive, white-box attack (Perfect-Knowledge): The most powerful threat model, we assume here the adversary has access to the detector and can mount an adaptive attack. To perform this attack, we construct a new loss function, and generate adversarial examples that both fool the classifier and also evade the detector.

The most difficult step in this attack is to construct a loss function that can be used to generate adversarial examples. In some cases, such a loss function might not be readily available. In other cases, one may exist, but it may not be well-suited to performing gradient descent over. It is of critical importance to choose a good loss function, and we describe how to construct such a loss function for each attack.

Construct a black-box attack (Limited-Knowledge): This attack is the most difficult for the adversary. We assume the adversary knows what type of defense is in place but does not know the detector's paramaters. This evaluation is only interesting if (a) the zero-knowledge attack failed to generate adversarial examples, and (b) the perfect-knowledge attack succeeded. If the strong attack alone succeeded, when the adversary was not aware of the defense, they could mount the same attack in this black-box case. Conversely, if the white-box attack failed, then a black-box attack will also fail (since the threat model is strictly harder).

In order to mount this attack, we rely on the transferability property: the attacker trains a substitute model in the same way as the original model, but on a separate training set (of similar size, and quality). The attacker can access substitute model's parameters, and performs a white-box attack on the substitute model. Finally, we evaluate whether these adversarial examples transfer to the original model.

When the classifier and detector are separate models, we assume the adversary has access to the classifier but not the detector (we are analyzing the increase in security by using the detector).

If the detector and classifier are not separable (i.e., the classifier is trained to also act as a detector), then to perform a fair evaluation, we compare the adversarial examples generated with black-box access to the (unsecured) classifier to adversarial examples generated with only black-box access to both the classifier and detector.

3 SECONDARY CLASSIFICATION BASED DETECTION

We now turn to evaluating the ten defenses. The first category of detection schemes we study build a second classifier which attempts to detect adversarial examples. Three of the approaches take this direction.

For the remainder of this subsection, define $F(\cdot)$ to be the classification network and $D(\cdot)$ to be the detection network. $F(\cdot)$ is defined as in Section 2.1 outputting a probability distribution over the 10 classes, and $D : \mathbb{R}^{w \cdot h \cdot c} \to (-\infty, \infty)$ represent the logits of the likelihood the instance is adversarial. That is, $\text{sigmoid}(D(x)) : \mathbb{R}^{w \cdot h \cdot c} \to [0, 1]$ represents the probability the instance is adversarial.

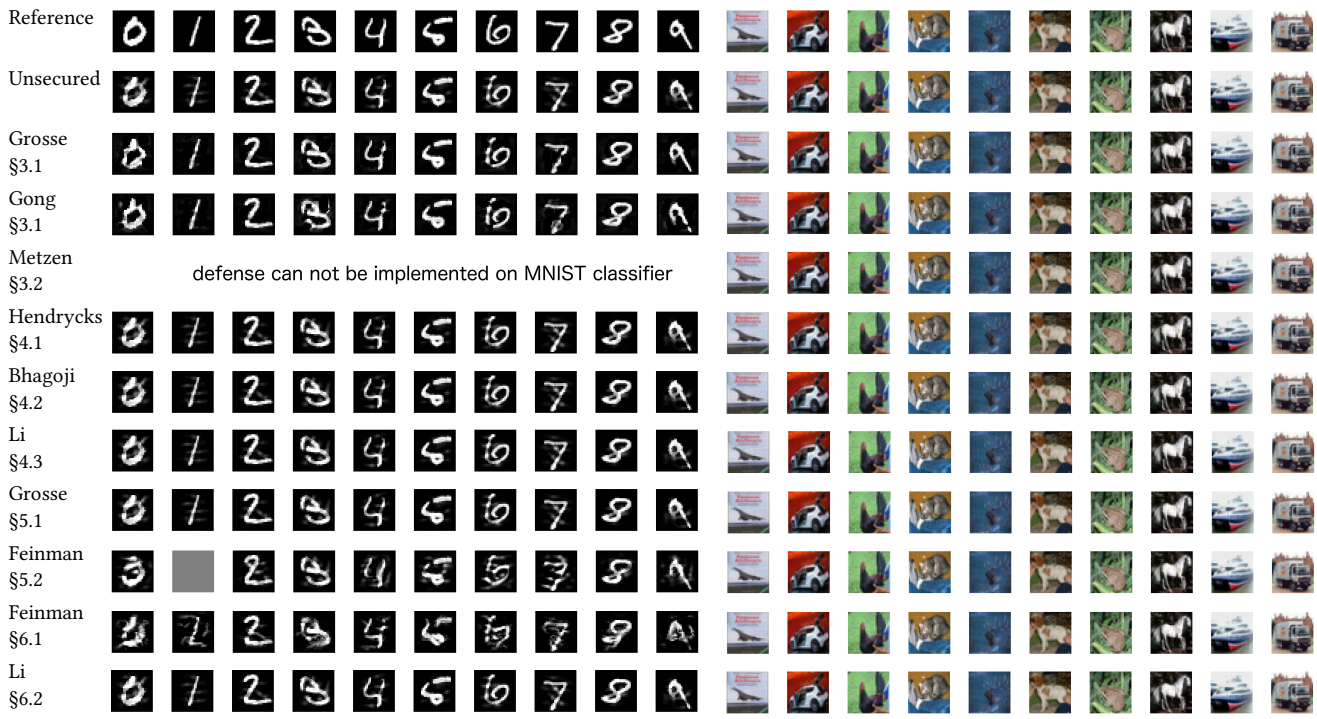

Figure 1: Summary of Results: adversarial examples on the MNIST and CIFAR datasets for each defense we study. The first row corresponds to the original images.

3.1 Adversarial Retraining

Grosse *et al.* [15] propose a variant on adversarial re-training. Instead of attempting to classify the adversarial examples correctly (by adding adversarial examples to the training set, with their correct labels), they introduce a new $N+1$st class — solely for adversarial examples — and train the network to detect adversarial examples. Specifically, they propose the following procedure:

(1) Train a model F_{base} on the training data $\mathcal{X}_0 = \mathcal{X}$.
(2) Generate adversarial examples on model F_{base} for each $(x_i, y_i) \in X$. Call these examples x'_i.
(3) Let $\mathcal{X}_1 = \mathcal{X}_0 \cup \{(x'_i, N+1) : i \in |\mathcal{X}|\}$ where $N+1$ is the new label for adversarial examples.
(4) Train a model F_{secured} on the training data $\mathcal{X}_1$.

Gong *et al.* [12] construct a very similar defense technique. Instead of re-training the model F_{secured} completely, they construct a binary classifier D that simply learns to partitions the instances x from x' by training on the training set

$$\mathcal{X}_1 = \{(x_i, 1) : i \in |\mathcal{X}|\} \cup \{(x'_i, 0) : i \in |\mathcal{X}|\}.$$

We re-implement these two defenses and find that adversarial retraining is able to detect adversarial examples when generated with the fast gradient sign and JSMA attacks with near-100% accuracy.

Zero-Knowledge Attack Evaluation. We train these two schemes on the entire MNIST training set, using C&W's attack to generate adversarial examples. In this way we construct a model F_{secured} and D.

We then construct adversarial examples for F_{base} from each image in the test set using C&W's attack. Both approaches detect these previously unseen test adversarial examples. Grosse *et al.* detects 98.5% of attacks as adversarial. Further, it classifies half of the remaining 1.5% correctly. Gong *et al.* achieve 98% accuracy in detecting adversarial examples.

Investigating further, we find that even if we train on adversarial examples generated using an *untargeted* attack, both schemes can detect *targeted* adversarial examples.

Neither of these defenses claim robustness against CIFAR, however when we perform this evaluation, we find to achieve a 70% detection rate requires a 40% false positive rate. This alone can be seen as a failure of these approaches on CIFAR.

Perfect-Knowledge Attack Evaluation. Next, we evaluate these defenses assuming the adversary is aware of these defenses and parameters. That is, we directly attack the defended model. Our experiments revealed that these defenses are ineffective and add almost no increase in robustness.

For Grosse's defense, we use C&W's attack on F_{secured} to generate adversarial examples; it succeeds 100% of the time. We computed the mean L_2-distance from the original sample to the adversarial example. Adversarial examples against F_{base} are at average L_2 distance of 2.05 from the original sample; adversarial examples against F_{secured} have an average distance of 2.26. Thus the defense has not reduced the success rate at generating adversarial examples, and has only increased the mean distortion by 10%.

Gong's defense does not fare any better. To help construct adversarial examples that will simultaneously fool F_{base} and D, we define a new function $G(\cdot)$ that represents the combination of the classifier (with logits $Z_F(\cdot)$) and detector (with logits $Z_D(\cdot)$). In particular, we define

$$G(x)_i = \begin{cases} Z_F(x)_i & \text{if } i \leq N \\ (Z_D(x)+1) \cdot \max_j Z_F(x)_j & \text{if } i = N+1 \end{cases}$$

Effectively, G acts as a classifier on $N+1$ classes. It is constructed to have two useful properties: if $Z_D(x) > 0$ (i.e., if the detector classifies x as malicious) then we will have

$$\arg\max_i(G(x)_i) = N+1$$

(where N is the new adversarial class), and if $Z_D(x) < 0$ (i.e., if the detector classifies x as natural) then we will have

$$\arg\max_i(G(x)_i) = \arg\max_i(Z_F(x)_i).$$

Why did we choose this particular function $G(\cdot)$? Recall from earlier that when using a gradient-descent based attack algorithm, there is a constant c in the loss function that trades off the relative importance of reducing the distance from the natural instance vs. classifying the input as the target class. If the value of the loss function changes by several orders of magnitude, no value of c will be optimal and adversarial examples will be very hard to find. Therefore, we construct G so that it is mostly linear.

Now we treat this function G as if it were a neural network, and feed it directly into C&W's attack algorithm instead of the function Z. It is easy to see that if an instance x' is classified by G as label $l \leq N$ then both $C(x') = l$ and the detector classifies x' as natural (non-adversarial). This approach successfully generates adversarial examples against Gong's defense with 100% success. The mean L_2 distortion of these adversarial examples is 2.31, an improvement of about 10% over the unprotected scheme.

We therefore conclude neither of these schemes are effective on MNIST. While adversarial examples are detectable with neural networks when not attempting to be evasive, it is possible to construct evasive adversarial examples.

On CIFAR, the above attacks work in an identical manner and give very similar results: the distortion is increased less than 5%.

Limited-Knowledge Attack Evaluation. For brevity, we only describe the attack on Grosse's scheme (Gong's results are identical). We generate transferable adversarial examples as described above. Instead of training one model on the complete training set, we split the training set in half. We assume the first half is known to the attacker and the second half is used by the defender to train their network. Thus, we construct two models R_1 and R_2 each trained on half of the training data. We treat R_2 as the defender's model (the target model for which we want to produce adversarial examples) and give the attacker white-box access to all parameters of model R_1. Then we apply the C&W attack to each instance in the test set to generate adversarial examples for R_1, and we test whether they fool R_2.

As a baseline, we applied this procedure to the unsecured model. The attack succeeded 98% of the time, with a mean L_2 distortion of 5.1. Then, we generated two secured models R_1 and R_2 implementing Grosse's defense, and applied the same attack. We observed a 98% success rate, with a mean L_2 distortion of 5.3, which is only 4% higher than the baseline. Therefore, we conclude that Grosse's defense is ineffective and can be attacked even by an attacker who does not the exact model parameters of the secured model.

3.2 Examining Convolutional Layers

In contrast to the prior approach, which attempts to detect adversarial examples based on the contents of the image itself, Metzen *et al.* [18] detect adversarial examples by looking at the inner convolutional layers of the network. They augment the classification neural network with a detection neural network that takes its input from various intermediate layers of the classification network. This detection network is trained identically to Gong's defense above. We refer interested readers to the original paper for complete details on the detector setup [18]. This defense only argues robustness against CIFAR, since it looks at the inner layers of a ResNet (which are not usually used for MNIST).

We then train the detector by attaching it to the output of the first residual block as done in [18]. We confirmed that their detector is able to detect adversarial examples with 99% probability on simple attacks (fast gradient sign or JSMA).

Zero-Knowledge Attack Evaluation. We found training the detector to be very difficult and highly sensitive; we experimented with different optimizers, hyper-parameters, and initializers to reduce the effect of this.

For evaluation, we generate adversarial examples for each instance in the test set, using the C&W attack. The best detector we were able to train correctly obtains an 81% true positive rate at 28% false positive rate. This is in stark contrast to the 99% success at detecting simpler attacks, which highlights that weak attacks give a biased view of defense robustness.

Similar to the prior defense (that also had a high false positive rate), this defense is unable to perform well even when the adversary is not attempting to evade it.

Perfect-Knowledge Attack Evaluation. Our white-box attack completely defeats Metzen's defense: it is able to produce adversarial examples that simultaneously are mis-classified by the original network and evade the detector. We generate adversarial examples using C&W's attack applied to the same function $G(\cdot)$ defined in Section 3.1. The mean distance to adversarial examples increases from 0.169 L_2 distortion on the unsecured model to 0.227 on the secured scheme, an improvement of 34%. However, in absolute terms, the adversarial examples generated are still indistinguishable from the original inputs.

Limited-Knowledge Attack Evaluation. To investigate if this defense is robust to attacks in a black-box setting, we perform a standard transferability test as done above. We split the training data in half, and train two detector models, one on each half of the training data. Then, we attack the second detector given only white-box access to the first detector.

On MNIST, we found that even low-confidence adversarial examples transfer 84% of the time between the two detectors when the classifier network is known by the adversary. By using high-confidence adversarial examples, the attack success rate can be increased to 98% at the cost of increasing the mean distortion by

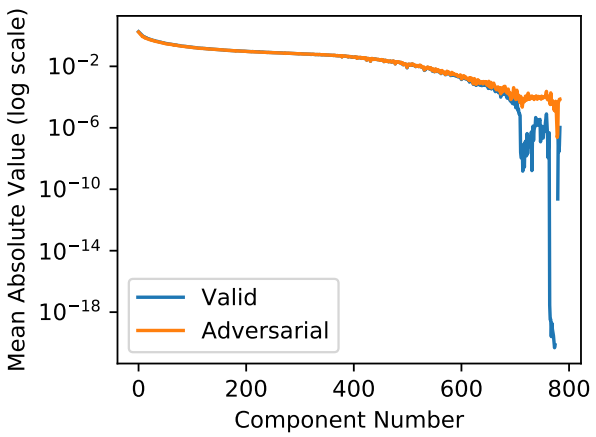

Figure 2: PCA on the MNIST dataset reveals a difference between natural images and adversarial images, however this is caused by an artifact of MNIST: border pixels on natural images are often 0 but slightly-positive on adversarial examples.

a further 28%, which is small enough that adversarial examples remain indistinguishable from the original images.

4 PRINCIPAL COMPONENT ANALYSIS DETECTION

Principal Component Analysis (PCA) transforms a set of points in a n-dimensional space to a new set of points in a k-dimensional space ($k \leq n$) through a linear transformation. We assume the reader is familiar with PCA for the remainder of this section.

4.1 Input Image PCA

Hendrycks & Gimpel [19] use PCA to detect natural images from adversarial examples, finding that adversarial examples place a higher weight on the larger principal components than natural images (and lower weight on the earlier principal components).

Zero-Knowledge Attack Evaluation. We first reproduce their results by running PCA on MNIST. To see if adversarial examples really do use larger principal components more often, we compute how much each component is used. Let $X_1, \ldots, X_n$ be the training set instances. We define the score $S(j)$ of the jth PCA component as

$$S(j) = \frac{1}{N} \sum_{i=1}^{N} |PCA(X_i)_j|.$$

We train a classification network on the training set and compute the component scores $S(1), \ldots, S(784)$. Then, for each image in the test set, we find the nearest adversarial example with C&W's attack and we compute the component scores on these adversarial examples. The results are plotted in Figure 2.

Our results agree with Hendrycks *et. al* [19]: there is no difference on the first principal components, but there is a substantial difference between natural and adversarial instances on the later components. On the MNIST data set, their defense does detect zero-knowledge attacks, if the attacker does not attempt to defeat the defense.

Looking Deeper. At first glance, this might lead us to believe that PCA is a powerful and effective method for detecting adversarial examples. However, whenever there are large abnormalities in the data, one must be careful to understand their cause.

In this case, the reason for the difference is that there are pixels on the MNIST dataset that are almost always set to 0. Since the MNIST dataset is constructed by taking 24x24 images and centering them (by center-of-mass) on a 28x28 grid, the majority of the pixels on the boundary of natural images are zero. Because these border pixels are essentially always zero for natural instances, the last principal components are heavily concentrated on these border pixels. This explains why the last 74 principal components (9.4% of the components) contribute less than 10^{-30} of the variance on the training set.

In short, the detected difference between the natural and adversarial examples is because the border pixels are nearly always zero for natural MNIST instances, whereas typical adversarial examples have non-zero values on the border. While adversarial examples are different from natural images on MNIST in this way, this is not an intrinsic property of adversarial examples; it is instead due to an artifact of the MNIST dataset. When we perform the above evaluation on CIFAR, there is no detectable difference between adversarial examples and natural data. As a result, the Hendrycks defense is not effective for CIFAR — it is specific to MNIST. Also, this deeper understanding of why the defense works on MNIST suggests that adaptive attacks might be able to avoid detection by simply leaving those pixels unchanged.

Perfect-Knowledge Attack Evaluation. We found that the Hendrycks defense can be broken by a white-box attacker with knowledge of the defense. Details are deferred to Section 4.2, where we break a strictly stronger defense. In particular, we found in our experiments that we can generate adversarial examples that are restricted to change only the first k principal components (i.e., leave all later components unchanged), and these adversarial examples that are not detected by the Hendrycks defense.

4.2 Dimensionality Reduction

Bhagoji *et al.* [4] propose a defense based on dimensionality reduction: instead of training a classifier on the original training data, they reduce the $W \cdot H \cdot C = N$-dimensional input (e.g., 784 for MNIST) to a much smaller K-dimensional input (e.g., 20) and train a classifier on this smaller input. The classifier uses a fully-connected neural network: PCA loses spatial locality, so a convolutional network cannot be used (we therefore consider only MNIST).

This defense restricts the attacker so they can only manipulate the first K components: the classifier ignores other components. If adversarial examples rely on the last principal components (as hypothesized), then restricting the attack to only the first K principal components should dramatically increase the required distortion to produce an adversarial example. We test this prediction empirically.

We reimplement their algorithm with their same model (a fully-connected network with two hidden layers of 100 units). We train 26

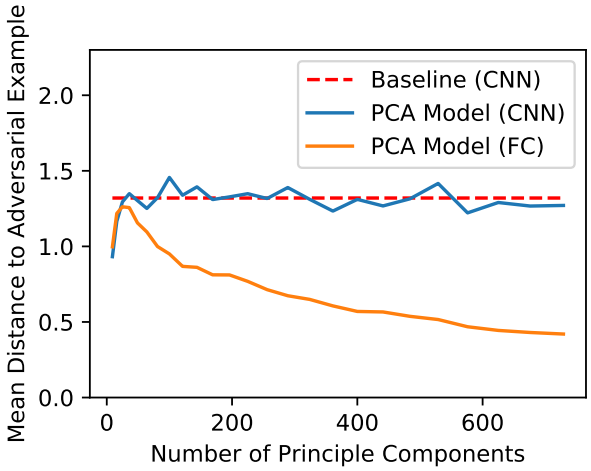

Figure 3: Performing dimensionality reduction increases the robustness of a 100-100-10 fully-connected neural network, but is still less secure than just using an unsecured CNN (the baseline). Dimensionality reduction does not help on a network that is already convolutional.

models with different values of K, ranging from 9 to 784 dimensions. Models with fewer than 25 dimensions have lower accuracy; all models with more than 25 dimensions have 97% or higher accuracy.

Perfect-Knowledge Attack Evaluation. We evaluate Bhagoji's defense by constructing targeted attacks against all 26 models we trained. We show the mean distortion for each model in Figure 3. The most difficult model to attack uses only the first 25 principal components; it is nearly 3× more robust than the model that keeps all 784 principal components.

However, crucially, we find that even the model that keeps the first 25 principal components is *less* robust than almost any standard, unsecured convolutional neural network; an unprotected network achieves both higher accuracy (99.5% accuracy) and better robustness to adversarial examples (measured by the mean distortion). In summary, Bhagoji's defense is not secure against white-box attacks.

Looking Deeper. Next, we show that this result is not an artifact of the network architecture — it is not caused just because fully-connected network are less robust than convolutional networks. We study a second algorithm that Bhagoji *et al.* present but did not end up using, which combines PCA with a convolutional neural network architecture. This allows us to perform an experiment where the network architecture is held fixed, and the only change is whether dimensionality reduction is used or not. In particular, instead of using the first K principal components as features for a fully-connected network, they use PCA to map each image into the reduced-dimensionality PCA space, and then immediately map it back to the image space. This effectively projects the image into a reduced-dimension manifold. They train a convolutional classifier on the projected images. This classifier achieves a higher accuracy (99% when using at least 25 dimensions).

In our experiments we found that this approach is no more robust than an unsecured convolutional network (applied to the original image), despite only using a limited number of the principal components. We conclude that defenses based on limiting the adversary to only the first principal components are not effective. It follows that Hendrycks *et. al*'s defense is broken as well, as the adversarial images generated in this way change only the first K components and leave all later components unchanged.

4.3 Hidden Layer PCA

Li *et al.* [24] apply PCA to the values after inner convolutional layers of the neural network, and use a cascade classifier to detect adversarial examples. Specifically, they propose building a *cascade classifier* that accepts the input as natural only if all classifiers C_i accept the input, but rejects it if any do. Each classifier C_i a linear SVM that acts on the PCA of the ith convolutional layer of the network.

They evaluate their scheme on ImageNet. In the remainder of this section we demonstrate their defense is not effective on MNIST and CIFAR; others have shown that attacking an ImageNet classifier would be even easier [8, 26].

Zero-Knowledge Attack Evaluation. Li *et al.* generated adversarial examples with Szegedy *et. al*'s L-BFGS algorithm [39] and found that the first linear SVM achieved 80% true positive rate at 0% false positive rate – an ideal use-case for a cascade classifier. We evaluated the effectiveness of their method at detecting adversarial examples generated with C&W's attack (when trained on adversarial examples generated with the C&W attack). For MNIST, their scheme only achieves a 62% true positive rate at a 37% false positive rate (and 56% at 40% on CIFAR) on the first layer of the cascade classifier. This results causes the cascade classifiers to have a 92% and 98% false positive rates on MNIST/CIFAR. We conclude that Li's defense is not effective against the C&W attack.

5 DISTRIBUTIONAL DETECTION

Next, we study two defenses that detect adversarial examples by comparing the distribution of natural images to the distribution of adversarial examples. They use classical statistical methods to distinguish natural images from adversarial images.

5.1 Maximum Mean Discrepancy

Grosse *et al.* [15] consider a very powerful threat model: assume we are given two sets of images S_1 and S_2, such that we know S_1 contains only natural images, and we know that S_2 contains either all adversarial examples, or all natural images. They ask the question: can we determine which of these two situations is the case?

To achieve this, they use the Maximum Mean Discrepancy (MMD) test [7, 14], a statistical hypothesis test that answers the question "are these two sets drawn from the same underlying distribution?"

The MMD is a theoretically useful technique that can be formally shown to always detect a difference if one occurs. However, it is computationally infeasible to compute, so a simple polynomial-time approximation is almost always used. In our experiments, we use the same approximation used by Grosse *et al.* [14].

To test whether X_1 and X_2 are drawn from the same distribution, Grosse *et al.* use Fisher's permutation test [28] with the MMD test statistic. To do this, initially let $a = MMD(X_1, X_2)$. Then, shuffle the elements of X_1 and X_2 into two new sets Y_1 and Y_2, and let $b = MMD(Y_1, Y_2)$. If $a < b$ then reject the null hypothesis, and conclude they are drawn from different distributions. Repeat this process many times, and repor the p-value as the fraction of times the null hypothesis is rejected.

As a first step, we verified that MMD correctly detects natural instances as being natural. To do this, we assign X_1 and X_2 to be random samples from the test set. The procedure above correctly fails to reject the null hypothesis ($p > 0.05$) in distinguishing the two distributions.

Zero-Knowledge Attack Evaluation. We repeated this experiment, this time producing targeted adversarial examples with C&W's attack algorithm. Even when using a set of 100 images, MMD fails to reject the null hypothesis ($p > 0.05$). MMD also fails to detect attacks on CIFAR, even with a set size as large as 10, 000 (which uses every sample in the test set). Since MMD is one of the most powerful multidimensional statistical tests, and even it is not effective, we argue that without significant modification, statistical tests will not be able to detect adversarial examples.

5.2 Kernel Density Estimation

Feinman *et al.* [11] present a defense they call *kernel density estimation*. They use a Gaussian Mixture Model to model outputs from the final hidden layer of a neural network, and argue that adversarial examples belong to a different distribution than that of natural images.

Specifically, given an instance x classified as label t, kernel density estimation estimates the likelihood of x as

$$KDE(x) = \frac{1}{|X_t|} \sum_{s \in X_t} \exp(\frac{|F^{n-1}(x) - F^{n-1}(s)|^2}{\sigma^2})$$

where X_t is the set of training instances with label t and $F^{n-1}(x)$ is the output of the final hidden layer on input x. The detector is therefore constructed by selecting a threshold τ and reporting x as adversarial if $KDE(x) < \tau$, otherwise reporting x as natural.

The motivation behind this approach is that the later hidden layers of a neural network have been shown to capture high-level semantic information about the input. Therefore, using a simple classifier on this final layer will be more accurate than if it were applied to the original input images, as the prior defense did.

Zero-Knowledge Attack Evaluation. Feinman's defense is able to detect C&W adversarial examples on MNIST, but not on CIFAR. Looking deeper, on CIFAR, for each image in the test set x and closest adversarial example x', we compare $KDE(x')$ to $KDE(x)$. Surprisingly, we find that 80% of the time, the adversarial example has a *higher* likelihood score than the original image.Therefore, Feinman's defense cannot work on CIFAR. In the remainder of this section, we show how to break this defense on MNIST with increased distortion.

Perfect-Knowledge Attack Evaluation. To mount a white-box attack, we construct a new minimization formulation that differs from the original only in that we introduce a new loss term $\ell_2(x')$ that penalizes being detected by the detector:

$$\text{minimize} \quad \|x - x'\|_2^2 + c \cdot (\ell(x') + \ell_2(x'))$$

where we define

$$\ell_2(x') = \max(-\log(KDE(x')) - \epsilon, 0)$$

where ϵ controls the likelihood measure of the adversarial examples. In our attack, we set ϵ to the median of $-\log(KDE(\cdot))$ on the training set, so that $\ell_2(x') \leq 0$ if and only if $KDE(x')$ is greater than half of the training instances KDE.

In practice, we mount this attack in two phases. First, we solve the original C&W minimization formulation to obtain an adversarial example $\hat{x}$. Typically $\hat{x}$ will be detected by the detector, so in the second phase we modify it to no longer be detected: we use this $\hat{x}$ as the initial value of x' in the above optimization problem and use gradient descent to improve it. Performing this two-step optimization is useful to allow for different constants c chosen for initially generating an adversarial example and for making it not detected.

This approach is effective at generating adversarial examples for MNIST that are not detected by Feinman's defense. The mean distortion is 3.81, up from 2.05 on an unsecured network. In order to implement this attack more efficiently, instead of computing the KDE across all training instances during the inner optimization loop, we randomly sample 100 instances (< 5% of the total) at first and use only those. A distortion of 3.81 begins to allow humans to notice the adversarial nature of these images (Figure 1).

We believe the reason this defense is so much more effective on MNIST than on CIFAR because this defense is effectively a K-nearest-neighbor classifier with a neural network feature extractor. In fact, prior work has shown that using an RBF kernel SVM (essentially identical to what is done here) is effective at defending MNIST to adversarial examples.

Limited-Knowledge Attack Evaluation. Again we only demonstrate the vulnerability on MNIST: the defense does not work on CIFAR. Note that our white-box attack algorithm didn't use the entire training set; it uses only a random sample of 100 instances from the training set (less than 5% of the training instances). This hints that black-box attacks might succeed. This is indeed the case: we found that when the defender uses the first 95% of the training data, and the last 5% of the training data is left for the adversary (who uses it to form their own KDE estimator), the attack still succeeds with equal distortion.

6 NORMALIZATION DETECTION

6.1 Dropout Randomization

Feinman *et al.* propose a second detection method called *Bayesian neural network uncertainty* that measures the uncertainty of the neural network on the given input. Instead of relying on the reported confidence of the network (which can easily be controlled by an adversary), they add randomization to the network. The hope is that a natural image will have the same (correct) label regardless of the random values chosen, while adversarial examples won't always be predicted with the same label. *Dropout* [37] is used as the method of adding randomness.

To determine the uncertainty of the network on a given instance, we apply the randomized network F_r (with dropout enabled) to the input L times. Then, the uncertainty is defined as

$$U(x) = \left(\frac{1}{L}\sum_{i=1}^{L} \|F_r(x)\|\right) - \left\|\frac{1}{L}\sum_{i=1}^{L} F_r(x)\right\|$$

where we take $\|y\|$ to mean $y^T y$ (the squared L_2 norm). Notice that if the predictions $F_r(x)$ are the same every time, the uncertainty is 0. In other words, the uncertainty is the sum of the variances of each component of the output, computed over the L outputs from the randomized network.

To classify an input x as adversarial or natural, we pick a threshold τ and a number of times to evaluate the network L and compute $U(x)$. We have found the algorithm is not sensitive to the choice of L as long as it is greater than 20 (and we pick $L = 30$ for the remainder of this section). We return "adversarial" if $U(x) > \tau$ and "natural" otherwise.

When we run this detection scheme on the adversarial examples generated by the FGS and JSMA attack algorithms on MNIST, it detects 96% of adversarial examples with a false positive rate lower than 1% when $\tau = 0.02$.

Zero-Knowledge Attack Evaluation. We generated adversarial examples with C&W's attack and find 75% of these adversarial examples are detected by this defense on MNIST. Surprisingly, this defense is even more effective at detecting attacks on CIFAR: it detects 95% of C&W adversarial examples.

Perfect-Knowledge Attack Evaluation. Our white-box attacks work by defining an objective function that combines the model and detector. In an initial attempt, we follow the approach taken to break the prior detection schemes, let

$$G(x)_i = \begin{cases} Z(x)_i & \text{if } i \le N \\ (1 + U(x) - \tau) \cdot \max_i Z(x)_i & \text{if } i = N+1 \end{cases}$$

(where $N+1$ is the new adversarial class). This objective function is defined so that if $U(x) > \tau$ (i.e., x is classified as adversarial by Feinman's defense) then we will have

$$\arg\max_i(G(x)_i) = N + 1,$$

and if $U(x) < \tau$ (i.e., x is classified as natural) then

$$\arg\max_i(G(x)_i) = \arg\max_i(F(x)_i).$$

We then attempt to generated adversarial examples by applying C&W's attack to $G(\cdot)$. However, this attack fails: the defense is still able to detect 60% of adversarial examples.

Looking Deeper. We investigate why this attack failed. Because values of the function $U(\cdot)$ are computed by calling $F(\cdot)$ (and not $Z(\cdot)$), there is no good constant c that C&W's algorithm can select. This causes gradient descent to perform poorly and rarely returns successful adversarial examples.

Therefore, for this defense, we abandon the approach of constructing a single function to optimize over and instead change the loss function ℓ used by the attack. Recall that the typical loss function used (for a given network $Z(\cdot)$) is

$$\ell_Z(x') = \max(\max\{Z(x')_i : i \neq t\} - Z(x')_t, -\kappa).$$

If every prediction $F_r(x)$ of the randomized network produced a very similar result, the uncertainty value $U(x)$ would be very small.

We sample K different deterministic networks $\{Z_j(\cdot) : j \in [1, K]\}$ each with different randomness used during dropout. If we were able to have $\arg\max_i Z_j(x)_i = t$ for every network j, for K big enough, it would be highly likely that $F_r(x)$ would always produce label t for any randomness. Thus, we construct a new loss function $\ell'(x') = \sum_{j=1}^{K} \ell_{Z_j}(x')$ as the average of the loss functions on each fixed model Z_j. Then we use C&W's attack with this revised loss function.

This approach successfully generates adversarial examples that fool the dropout defense with 98% success. On MNIST, the mean l_2 distortion is 3.68, up from the baseline of 2.05. This is the largest distortion required by any of the defenses we have evaluated; the distortion here is nearing the levels of human perceptibility (Figure 1). On CIFAR the distortion required again increases to 1.1, a factor of 5× larger, but is still entirely imperceptible (Figure 1).

Limited-Knowledge Attack Evaluation. It turns out that adversarial examples generated with the white-box approach transfer with high probability across models. This is due to the fact that our white-box attack assumes we do not know the exact randomization settings, and therefore construct adversarial examples that are effective regardless of randomization. This is similar to the black-box threat model, where the adversary does not have access to the model parameters.

However, to improve the rate of transferability, we again construct two models $F(\cdot)$ and $G(\cdot)$ on different subsets of the training data. We provide the adversary access to the parameters of F and use the white-box attack above to generate adversarial examples for F; we then test whether they fool G.

On MNIST, we find that adversarial examples generated with our algorithm transfer to G with 90% success. We can increase the transfer rate to 98% at the cost of increasing the mean distortion only 15%, to 4.23. While this 15% increase may seem like a failure of this defense under the black-box threat model, we see this instead as a success of defense under the white-box threat model. It has made constructing adaptive white-box attacks nearly as hard as constructing black-box attacks.

In fact, we find that this is the reason why the CIFAR network has such a larger increase in distortion: to obtain transferable cifar adversarial examples requires a distortion of approximately 4× larger than non-transferable attacks. We consider this the only defense we evaluate that is not completely broken (even though we still can generate adversarial examples that are imperceptible on CIFAR).

6.2 Mean Blur

The second detection method proposed by Li *et al.* applies a 3×3 average filter to blur the image before applying the classifier. The authors admit this defense is "overly simplistic" but still argue it is effective at alleviating adversarial examples. We confirm this simple defense can remove adversarial examples generated with fast gradient sign, as they found in their paper.

Zero-Knowledge Attack Evaluation. When we use C&W's attack, we find that this defense effectively removes low-confidence adversarial examples: 80% of adversarial examples (at a mean L_2 distortion of 2.05) are no longer classified incorrectly.

This attack can even partially alleviate high-confidence adversarial examples. To ensure they remain adversarial after blurring, we must increase the distortion by a factor of 3×.

Perfect-Knowledge Attack Evaluation. Observe that taking the mean over every 3×3 region on the image is the same as adding another convolutional layer to the beginning of the neural network with one output channel that performs this calculation. Given the network F, we define $F'(x) = F(\text{blur}(x))$ and apply C&W's attack against F'. When we do so, we find that the mean distance to adversarial examples does not increase. Therefore, blurring is not an effective defense.

7 LESSONS

7.1 Properties of adversarial examples

After examining these ten defenses, we now draw conclusions about the nature of the space of adversarial examples and the ability to detect them with different approaches.

Randomization can increase required distortion. By far the most effective defense technique, dropout randomization, made generating adversarial examples nearly five times more difficult on CIFAR. In particular, it makes generating adversarial examples on the network as difficult as generating transferable adversarial examples, a task known to be harder [29]. Additionally, if it were possible to find a way to eliminate transferability, a randomization-based defense may be able to detect adversarial examples. At this time, we believe this is the most promising direction of future work.

MNIST properties may not hold on CIFAR. Most defenses that increased the distortion on MNIST had a significantly lower distortion increase on CIFAR. In particular, kernel density estimation, the most effective defense on MNIST, was completely ineffective on CIFAR.

Detection neural networks can be bypassed. Across all of the defenses we evaluate, the least effective schemes used another neural network (or more neural network layers) to attempt to identify adversarial examples. Given that adversarial examples can fool a single classifier, it makes sense that adversarial examples can fool a classifier and detector.

Operating on raw pixel values is ineffective. Defenses that operated directly on the pixel values were too simple to succeed. On MNIST, these defenses provided reasonable robustness against weak attacks; however when evaluating on stronger attacks, these defenses all failed. This should not be surprising: the reason neural networks are used is that they are able to extract deep and meaningful features from the input data. A simple linear detector is not effective at classification when operating on raw pixel values, so it should not be surprising it does not work at detecting adversarial examples. (This can be seen especially well on CIFAR, where even weak attacks often succeed against defenses that operate on the input pixel space.)

7.2 Recommendations for Defenses

We have several recommendations for how researchers proposing new defenses can better evaluate their proposals. Many of these recommendations may appear to be obvious, however most of the papers we evaluate do not follow any.

Evaluate using a strong attack. Evaluate proposed defenses using the strongest attacks known. *Do not use fast gradient sign or JSMA exclusively*: most defenses that detect these attacks fail against stronger attacks. In particular, Fast gradient sign was not even designed to produce high-quality attacks: it was created to demonstrate neural networks are highly linear. Using these algorithms as a first test is reasonable first step, but is not sufficient. We recommend new schemes evaluate against strong iterative attacks.

Demonstrate white-box attacks fail. It is not sufficient to show that a defense can detect adversarial examples: one must also show that a adversary aware of the defense can not generate attacks that evade detection. We show how to perform that kind of evaluation: construct a differentiable function that is minimized when the image fools the classifier and is treated as natural by the detector, and apply a strong iterative attack (e.g., C&W's attack) to this function.

Report false positive and true positive rates. When constructing a detection-based defense, it is not enough to report the accuracy of the detector. A 60% accuracy can either be very useful (e.g., if it achieves a high true-positive rate at a 0% false-positive rate) or entirely useless (e.g., if it detects most adversarial images as adversarial at the cost of many natural images as adversarial). Instead, report both the false positive and true positive rates. To allow for comparisons with other work, we suggest reporting at least the true positive rate at 1% false positive rate; showing a ROC curve would be even better.

Evaluate on more than MNIST. We have found that defenses that only evaluated on the MNIST dataset typically either (a) were unable to produce an accurate classifier on CIFAR, (b) were entirely useless on CIFAR and were not able to detect even the fast gradient sign attack, or (c) were even weaker against attack on CIFAR than the other defenses we evaluated. Future schemes need to be evaluated on multiple data sets — evaluating their security solely on MNIST is not sufficient. While we have found CIFAR to be a reasonable task for evaluating security, in the future as defenses improve it may become necessary to evaluate on harder datasets (such as ImageNet [10]).

Release source code. In order to allow others to build on their work, authors should release the source code of their defenses. Not releasing source code only sets back the research community and hinders future security analysis. Seven of the ten we evaluate did not release their code (even after contacting the authors), requiring us to reimplement the defenses before evaluation.

8 CONCLUSION

Unlike standard machine-learning tasks, where achieving a higher accuracy on a single benchmark is in itself a useful and interesting result, this is not sufficient for secure machine learning. We must consider how an attacker might react to any proposed defense, and

evaluate whether the defense remains secure against an attacker who knows how the defense works.

In this paper we evaluate ten proposed defenses and demonstrate that none of them are able to withstand a white-box attack. We do this by constructing defense-specific loss functions that we minimize with a strong iterative attack algorithm. With these attacks, on CIFAR an adversary can create imperceptible adversarial examples for each defense.

By studying these ten defenses, we have drawn two lessons: existing defenses lack thorough security evaluations, and adversarial examples are much more difficult to detect than previously recognized. We hope that our work will help raise the bar for evaluation of proposed defenses and perhaps help others to construct more effective defenses. Further, our evaluations of these defenses expand on what is believed to be possible with constructing adversarial examples: we have shown that, so far, there are no known intrinsic properties that differentiate adversarial examples from regular images. We believe that constructing defenses to adversarial examples is an important challenge that must be overcome before these networks are used in potentially security-critical domains, and hope our work can bring us closer towards this goal.

9 ACKNOWLEDGEMENTS

We would like to thank Kathrin Grosse, Fuxin Li, Reuben Feinman, Metzen Jan Hendrik for discussing their defenses with us, and the anonymous reviewers for their feedback. This work was supported by the AFOSR under MURI award FA9550-12-1-0040, Intel through the ISTC for Secure Computing, the Hewlett Foundation through the Center for Long-Term Cybersecurity, and Qualcomm.

REFERENCES

[1] Marco Barreno, Blaine Nelson, Anthony D Joseph, and JD Tygar. 2010. The security of machine learning. *Machine Learning* 81, 2 (2010), 121–148.

[2] Marco Barreno, Blaine Nelson, Russell Sears, Anthony D Joseph, and J Doug Tygar. 2006. Can machine learning be secure?. In *Proceedings of the 2006 ACM Symposium on Information, computer and communications security*. ACM, 16–25.

[3] Osbert Bastani, Yani Ioannou, Leonidas Lampropoulos, Dimitrios Vytiniotis, Aditya Nori, and Antonio Criminisi. 2016. Measuring neural net robustness with constraints. In *Advances In Neural Information Processing Systems*. 2613–2621.

[4] Arjun Nitin Bhagoji, Daniel Cullina, and Prateek Mittal. 2017. Dimensionality Reduction as a Defense against Evasion Attacks on Machine Learning Classifiers. *arXiv preprint arXiv:1704:02654* (2017).

[5] Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Šrndić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. 2013. Evasion attacks against machine learning at test time. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 387–402.

[6] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, and others. 2016. End to End Learning for Self-Driving Cars. *arXiv preprint arXiv:1604.07316* (2016).

[7] Karsten M Borgwardt, Arthur Gretton, Malte J Rasch, Hans-Peter Kriegel, Bernhard Schölkopf, and Alex J Smola. 2006. Integrating structured biological data by kernel maximum mean discrepancy. *Bioinformatics* 22, 14 (2006), e49–e57.

[8] Nicholas Carlini and David Wagner. 2017. Towards evaluating the robustness of neural networks. *IEEE Symposium on Security and Privacy* (2017).

[9] Nilesh Dalvi, Pedro Domingos, Sumit Sanghai, Deepak Verma, and others. 2004. Adversarial classification. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 99–108.

[10] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE, 248–255.

[11] Reuben Feinman, Ryan R Curtin, Saurabh Shintre, and Andrew B Gardner. 2017. Detecting Adversarial Samples from Artifacts. *arXiv preprint arXiv:1703.00410* (2017).

[12] Zhitao Gong, Wenlu Wang, and Wei-Shinn Ku. 2017. Adversarial and Clean Data Are Not Twins. *arXiv preprint arXiv:1704.04960* (2017).

[13] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572* (2014).

[14] Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. 2012. A kernel two-sample test. *Journal of Machine Learning Research* 13, Mar (2012), 723–773.

[15] Kathrin Grosse, Praveen Manoharan, Nicolas Papernot, Michael Backes, and Patrick McDaniel. 2017. On the (Statistical) Detection of Adversarial Examples. *arXiv preprint arXiv:1702.06280* (2017).

[16] Shixiang Gu and Luca Rigazio. 2014. Towards deep neural network architectures robust to adversarial examples. *arXiv preprint arXiv:1412.5068* (2014).

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 770–778.

[18] Jan Hendrik Metzen, Tim Genewein, Volker Fischer, and Bastian Bischoff. 2017. On Detecting Adversarial Perturbations. In *International Conference on Learning Representations*. arXiv preprint arXiv:1702.04267.

[19] Dan Hendrycks and Kevin Gimpel. 2017. Early Methods for Detecting Adversarial Images. In *International Conference on Learning Representations (Workshop Track)*.

[20] Ruitong Huang, Bing Xu, Dale Schuurmans, and Csaba Szepesvári. 2015. Learning with a strong adversary. *CoRR, abs/1511.03034* (2015).

[21] Jonghoon Jin, Aysegul Dundar, and Eugenio Culurciello. 2015. Robust Convolutional Neural Networks under Adversarial Noise. *arXiv preprint arXiv:1511.06306* (2015).

[22] Alex Krizhevsky and Geoffrey Hinton. 2009. Learning multiple layers of features from tiny images. (2009).

[23] Yann LeCun, Corinna Cortes, and Christopher JC Burges. 1998. The MNIST database of handwritten digits. (1998).

[24] Xin Li and Fuxin Li. 2016. Adversarial Examples Detection in Deep Networks with Convolutional Filter Statistics. *arXiv preprint arXiv:1612.07767* (2016).

[25] Daniel Lowd and Christopher Meek. 2005. Adversarial learning. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*. ACM, 641–647.

[26] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. 2016. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2574–2582.

[27] Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*. 807–814.

[28] Anders Odén and Hans Wedel. 1975. Arguments for Fisher's permutation test. *The Annals of Statistics* (1975), 518–520.

[29] Nicolas Papernot, Patrick McDaniel, and Ian Goodfellow. 2016. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *arXiv preprint arXiv:1605.07277* (2016).

[30] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. 2016. The limitations of deep learning in adversarial settings. In *Security and Privacy (EuroS&P), 2016 IEEE European Symposium on*. IEEE, 372–387.

[31] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. 2016. Distillation as a defense to adversarial perturbations against deep neural networks. *IEEE Symposium on Security and Privacy* (2016).

[32] Slav Petrov. 2016. Announcing syntaxnet: The world's most accurate parser goes open source. *Google Research Blog, May* 12 (2016), 2016.

[33] Andras Rozsa, Ethan M Rudd, and Terrance E Boult. 2016. Adversarial diversity and hard positive generation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 25–32.

[34] Uri Shaham, Yutaro Yamada, and Sahand Negahban. 2015. Understanding Adversarial Training: Increasing Local Stability of Neural Nets through Robust Optimization. *arXiv preprint arXiv:1511.05432* (2015).

[35] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, and others. 2016. Mastering the game of Go with deep neural networks and tree search. *Nature* 529, 7587 (2016), 484–489.

[36] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. 2015. Striving for simplicity: The all convolutional net. In *International Conference on Learning Representations (Workshop Track)*.

[37] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research* 15, 1 (2014), 1929–1958.

[38] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2818–2826.

[39] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2014. Intriguing properties of neural networks. (2014).

[40] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, and others. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144* (2016).

[41] Stephan Zheng, Yang Song, Thomas Leung, and Ian Goodfellow. 2016. Improving the robustness of deep neural networks via stability training. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 4480–4488.