Adversarial Examples Are Not Easily Detected: Bypassing Ten Detection Methods

Nicholas Carlini
David Wagner

ABSTRACT

Neural networks are known to be vulnerable to adversarial examples: inputs that are close to valid inputs but classified incorrectly. We investigate the security of ten recent proposals that are designed to detect adversarial examples. We show that *all* can be defeated, even when the adversary does not know the exact parameters of the detector. We conclude that adversarial examples are significantly harder to detect than previously appreciated, and we propose several 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 [35], to beating the best humans at Go [32], to translation and NLP [29], and to self driving cars [3, 5].

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 turn out to be weak against an adversary who attempts to fool the classifier. That is, given a valid 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 [36], 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 [1, 13, 17, 19, 28, 30, 31, 37]. 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 [2, 8, 9, 12, 15, 16, 23], and 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 are 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 valid images. We also use this experimentation to obtain a better understanding of the space of adversarial examples.

We introduce three types of attacks in this paper. First, we evaluate the security of these schemes against generic attacks that don't take any specific measures to fool any particular detector. We use methods to generate *high-confidence adversarial examples*: instead of generating adversarial examples to minimize the total amount of distortion required to *just* cross the decision boundary, we construct adversarial examples that are classified as an incorrect class, with high confidence scores, but still have low distortion. 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, which are tailored for defeating a particular defense. These attacks require the adversary to have knowledge of the defense's model parameters. 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. Our white-box attacks can break all of the defenses.

Finally, we introduce black-box attacks, which are tailored towards a particular defense but do not require knowledge of the defense's model parameters. These attacks work by exploiting transferability [36], and our techniques for generating adversarial examples that transfer well may be of independent interest. We again show that all of the defenses can be broken in this way.

Our results suggest that there is a need for better ways to evaluate potential defenses. We believe our attacks would be a useful baseline: to be worth considering, a proposed defense must at least defeat the attacks described here.

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 high-confidence adversarial examples, even when the attacker is oblivious to the specific defense used.
- We break all existing detection methods in the white-box setting.
- We also show how to extend our attacks to the black-box setting, where the adversary does not have access to the defense model parameters.
- We provide recommendations for evaluating future defenses.

2 BACKGROUND

The remainder of this section contains a brief survey of the field of neural networks and adversarial machine learning. We encourage readers unfamiliar with this area to read the following papers (in this order): [36], [10], [26], and [6].

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

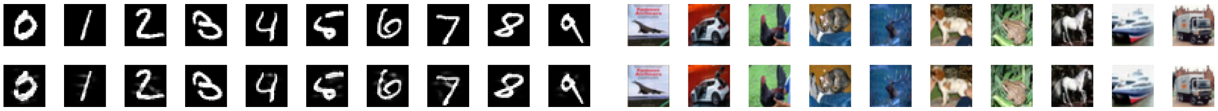

Figure 1: The first row contains images from the MNIST and CIFAR-10 datasets, chosen as the first image of each of the 10 classes in the test set. The second row shows untargeted adversaries examples generated by Carlini and Wagner's attack algorithm for the L_2 distance metric. Adversarial examples on MNIST have low distortion; distortion on CIFAR is $10\times$ smaller.

layers. The outputs of the final layer are known as logits; we represent them by $Z(\cdot)$. Some layers involve the non-linear ReLU [25] 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

We call an input to the classifier $F(\cdot)$ *valid* if it is instance that was benignly created. All instances in the training set and testing set are valid instances.

Given a network $F(\cdot)$ and a valid 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.[1] 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 [27]; L_2, the standard Euclidean norm [6, 24, 36]; or L_∞, a measure of the maximum absolute change to any pixel [10].[2] If the total distortion under any of these three distance metrics is small, the images will likely appear visually similar.

One further property of adversarial examples we will make use of is the transferability property [10, 36]. 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 [6, 10, 24, 27] and to make networks robust against adversarial examples [1, 13, 17, 19, 28, 30, 31, 37]. 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 [2, 8, 9, 12, 15, 16, 23].

2.3 Threat Model

We consider three different threat models in this paper:

(1) An *Oblivious 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 *White-Box 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 *Black-Box 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 3.

False Positive vs. False Negative Tradeoff. In order to be useful in practice, an adversarial detection scheme must be able to support a very low false-positive rate while still detecting adversarial examples, since false positives correspond directly to a decrease in accuracy. The trivial defense that reports every input as being an adversarial input will have a 100% true positive rate, but also will be entirely useless.

In this paper, all of our attacks are constructed in such a way that the defense can detect our attack with probability no better than random guessing. Specifically, we assume the defender is willing to tolerate a false-positive rate as high as 50%, and we show attacks that reduce the true positive rate to only 50%, equivalent to random guessing.

In practice, a defender would probably need the false positive rate to be well below 1%, and an attacker might be satisfied with an attack that succeeds with probability well under 50%. Therefore, our attacks go well beyond what would be needed to break a scheme; they show that the defenses we analyze are not effective.

2.4 Datasets

We consider two datasets in this paper.

[1] Untargeted adversarial examples are only interesting if the network predicted the instance correctly initially; targeted adversarial examples are only interesting if the target class is not the correct class.

[2] L_0 is defined as the limit of L_δ as $\delta \rightarrow 0$ (and similarly for L_∞).

Paper	Classification	PCA	Distributional	Normalization
Metzen *et al.* [15]	NA/34% (§4.2)			
Gong *et al.* [9]	11%/24% (§4.1)			
Grosse *et al.* [12]	10%/8% (§4.1)		0%/0% (§6.1)	
Feinman *et al.* [8]			85%/0% (§6.2)	85%/95% (§7.1)
Li *et al.* [23]		0%/0% (§5.3)		0%/0% (§7.2)
Bhagoji *et al.* [2]		0%/0% (§5.2)		
Hendrycks *et al.* [16]		0%/0% (§5.1)		

Table 1: Summary of our results. We break all ten schemes. The table shows the increase in the distortion of adversarial examples generated by our attack, compared to the distortion of adversarial examples on an unprotected network, on MNIST / CIFAR. 0% implies the defense adds no value. See Figure 6 and 7 (appendix) for examples of adversarial images we generated.

The **MNIST** dataset [22] 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 [20] 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 [33]. For comparison with prior work, we use the CIFAR-10 architecture from Metzen *et al.* [15]. The model is a 32-layer ResNet [14] with 470k parameters that we train with SGD with momentum set to 0.9. The learning rate is initially set to 0.1 and is reduced to 0.01 at epoch 20, and further reduced to 0.001 at epoch 40. We train with dataset augmentation, randomly flipping the image horizontally, and shifting by 3 pixels in any dimension. We apply L_2 regularization and use BatchNorm [18] after every convolution. This model achieves a 91.5% accuracy.

In Figure 1 we show images from each of these datasets, as well as the nearest untargeted adversarial example under the L_2 distance metric. Notice that the total distortion required to change classification on MNIST is much larger than that of CIFAR: because CIFAR images are harder to classify, it is easier to fool the classifier into misclassifying them.

Some of the defenses we evaluate also argue robustness against ImageNet [7], a database of a million 224×224 images. Prior work [6, 24] has clearly demonstrated that constructing adversarial examples on ImageNet is a strictly easier task than MNIST or CIFAR, and constructing defenses is strictly harder. As a simple comparison, an ImageNet classification can be changed by only flipping the lowest bit of each pixel, whereas CIFAR requires distortions 3× larger (and MNIST 10×). We therefore do not consider ImageNet in this paper, since all defenses are broken on MNIST and CIFAR.

2.5 Defenses

We briefly describe the ten proposed defenses described in the seven papers we study.

(1) Grosse *et al.* [12] propose two schemes. The first uses a high-dimensional statistical test (Maximum Mean Discrepancy) to detect adversarial examples. The second trains the neural network with a new "adversarial" class.
(2) Gong *et al.* [9] detect adversarial examples by building a second neural network that detects adversarial examples from valid images.
(3) Metzen *et al* [15] follow a similar approach, but train the detector on the inner layers of the classifier.
(4) Li *et al.* [23] propose two schemes. The first performs PCA on the internal convolutional layers of the primary network and trains classifier to distinguish between valid and adversarial data. The second scheme applies a mean-blur to images before feeding them to the network.
(5) Hendrycks & Gimpel [16] perform PCA on the pixels of an image and argue adversarial examples place higher emphasis on larger components.
(6) Feinman *et al.* [8] detect adversarial examples by keeping dropout [34] on during evaluation; additionally, they construct a kernel density measure and show that adversarial examples are drawn from a different distribution than valid images.
(7) Bhagoji *et al.* [2] show that adversarial images require use of more PCA dimensions than valid images.

We summarize our results in Table 1, and Figure 6 and 7 (appendix) for images of adversarial examples on the MNIST and CIFAR datasets, when the defense success was greater than 0%.

2.6 Generating Adversarial Examples

We use the L_2 attack algorithm of Carlini and Wagner [6] to generate targeted adversarial examples, as it is superior to other published attacks. Given a neural network F with logits Z, the attack uses gradient descent to solve

$$\text{minimize} \quad \|\frac{1}{2}(\tanh(w) + 1) - x\|_2^2 + c \cdot \ell(\frac{1}{2}(\tanh(w) + 1))$$

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.

Aside from C&W's attack, there are two other weaker attacks that are used throughout the literature to evaluate defenses. As we will see throughout this paper, these attacks are easy to detect and we recommend that future work does not evaluate against them.

- The **Fast Gradient Sign** attack [10] takes a single step, for all pixels, in the direction of the gradient. This attack aims to optimize the L_∞ distance metric and is very efficient to implement.
- **JSMA** [27] is an attack that greedily modifies one pixel at a time until the image is classified incorrectly. At each iteration the pixel chosen is the one that maximizes a a specially-crafted loss function. This attack seeks to optimize the L_0 distance metric.

3 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: In this step we generate adversarial examples with C&W's attack and check whether the defense can detect this strong (but oblivious) 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: 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 based on the loss function used earlier, and generate adversarial examples that are targeted to both fool the classifier and also evade the detector.

The most difficult step in this attack is to construct a differentiable loss function that can be used to generate adversarial examples. In some cases, such a loss function might not be readily available (if, for example, the defense is not a differentiable function). In other cases, one may exist, but it may not be well-suited to performing gradient descent over. We describe how we construct such a loss function for each attack.

Construct a black-box attack: In the final, and most difficult, threat model, we assume the adversary knows what defense is in place but does not know the detector's paramaters. This evaluation is only interesting if (a) the strong attack failed to generate adversarial examples, and (b) the adaptive white-box 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: we as the attacker train a substitute model in the same way as the original model, but on a separate training set. We have access to the substitute model's parameters, so we perform a white-box attack on the substitute to generate high-confidence adversarial examples. Finally, we evaluate whether these adversarial examples transfer to the actual 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.

As with any experiment, it is better to change only one variable at a time: if we were to compare white-box-no-detector distortion to black-box-with-detector distortion, we would not know if the detector has added any robustness, or if it just that adversarial examples require larger distortion to transfer (which they do [6, 21]).

4 SECONDARY CLASSIFICATION BASED DETECTION

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 $D : \mathbb{R}^{w \cdot h \cdot c} \rightarrow (-\infty, \infty)$ represent the logits of the liklehood the instance is adversarial. That is, $\text{sigmoid}(D(x)) : \mathbb{R}^{w \cdot h \cdot c} \rightarrow [0, 1]$ represents the probability the instance is adversarial.

4.1 Adversarial Retraining

Grosse *et al.* [12] 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.* [9] construct a similar detector; however, instead of re-training the full network to detect the original examples along with adversarial examples, they train a second binary classifier to predict whether the instance is adversarial or valid. That is, they do the following:

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 $\mathcal{X}_1 = \{(x_i, 1) : i \in |\mathcal{X}|\} \cup \{(x'_i, 0) : i \in |\mathcal{X}|\}$.
(4) Train a model D on the training data $\mathcal{X}_1$.

For the remainder of this section we report results on the MNIST dataset; neither paper claims robustness on CIFAR (a strictly harder task). When we perform the attacks in this section on CIFAR, our results only improve.

Since we were unable to obtain source code for the defense of Grosse *et al.*, we re-implement it and confirm their results papers: adversarial retraining is able to detect adversarial examples when generated with the fast gradient sign and JSMA attacks with near-100% accuracy.

Oblivious 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} / 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 detect *targeted* adversarial examples (for any target class). Additionally, we find that both detection schemes when trained on low-confidence adversarial examples are able to detect high-confidence adversarial examples.

White-box Attack Evaluation. Next, we evaluate these defenses assuming the adversary is aware of these defenses and the model parameters. That is, we directly attack the defended model. Our experiments revealed that these defenses are ineffective and add no robustness. We can produce adversarial examples with 100% success for both.

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

$$G(x)_i = \begin{cases} Z_F(x)_i & \text{if } i \le N \\ (Z_D(x) + 1) \cdot \max\limits_i Z_F(x)_i & \text{if } i = N + 1 \end{cases}$$

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 valid) 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 valid 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 $l \le N$ then both $C(x') = l$ and the detector classifies x' as valid (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.

Thus, Gong's scheme performs about the same as Grosse's defense: neither is effective. In retrospect, it should not be surprising that the robustness increase on these two defenses is nearly identical – they are very similar.

Black-Box Evaluation. Next, we consider black-box attacks, where the parameters of the defended network are not known to the adversary. For brevity, we only describe the attack on Grosse's scheme (Gong's results are identical). We generate transferable adversarial examples as described in Section 3: 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.

4.2 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.* [15] 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 by generating adversarial examples for the classification network. We refer interested readers to the original paper for complete details on the detector setup [15].

We copy their network architecture exactly and reproduce their results as described. Our ResNet-32 architecture obtains 91.5% accuracy on CIFAR-10. We confirmed that their detector is able to detect adversarial examples with 99% probability on simple attacks (e.g., fast gradient sign or JSMA). Their paper demonstrates that it is also able to detect adversarial examples generated by two other attacks, DeepFool [24] and the Basic Iterative Method [21].

Oblivious Attack Evaluation. We take each instance in the training set and find the nearest adversarial example using the C&W attack. We then train the detector by attaching it to the output of the first residual block as done in [15]. 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 predicts 72% of the test data as valid, and 81% of the adversarial examples as adversarial. 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. This defense is unable to perform well even when the adversary is not attempting to evade it.

We experiment with several other approaches to increase the accuracy of the detector, including increasing the number of model parameters, adding additional regularization, placing the detector at different layers in the network, and applying an ensemble of multiple detectors placed at different layers in the network. None of these variants work substantially better.

We do not evaluate Metzen's defense on MNIST, because the defense is only applicable to ResNet architectures, which dramatically over-fit on MNIST.

White-box 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 4.1. 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.

Black-Box Evaluation. To investigate if this defense is robust to attacks in a black-box setting, we perform our standard transferability test. 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.

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

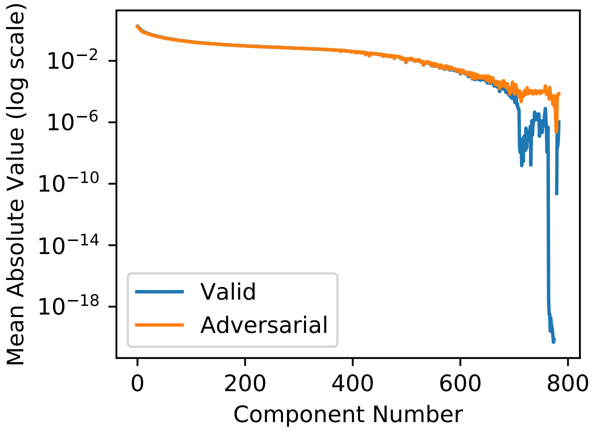

Figure 2: PCA on the MNIST dataset reveals a difference between valid images and adversarial images. An artifact of the MNIST dataset causes this: the outer pixels are almost always zero in valid images, but are sometimes non-zero in adversarial images.

5 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.

5.1 Input Image PCA

Hendrycks & Gimpel [16] use PCA to detect valid images from adversarial examples, finding that adversarial examples place a higher weight on the larger principal components than valid images (and lower weight on the earlier principal components).

Oblivious 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* [16]: there is no difference on the first principal components, but there is a substantial difference between valid and adversarial instances on the later components. On the MNIST data set, their defense does detect oblivious 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 the image are always zero. Because of this, the top two rows of pixels are all zero 99.9% of the time. The same is true for the left two columns and right two columns. The bottom row is zero 99% of the time. Because these border pixels are essentially always zero for valid instances, the last principal components are heavily concentrated on these border pixels.

Due to this effect, the last 74 principal components (9.4% of the components) explain less than 10^{-30} of the variance on the training set. (This is non-zero due exclusively to floating-point rounding errors.) When we look only at the first 710 principal components, we see a much less dramatic difference, and only a slight difference on components 700 to 710.

In short, the difference between valid and adversarial examples is because the border pixels are nearly always zero for valid MNIST instances, while existing attack algorithms often change the border pixels to non-zero values. While adversarial examples are different from valid images on MNIST, the reason is not an intrinsic property of adversarial examples and 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 valid 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 smarter attacks might be able to avoid detection by simply leaving those pixels unchanged.

White-Box and Black-Box Evaluation. We found that the Hendrycks defense can be broken by a white-box attacker with knowledge of the defense, and even by a black-box attacker. Details are deferred to Section 5.2, 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.

5.2 Dimensionality Reduction

Bhagoji *et al.* [2] 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.

This defense restricts the attacker so they can only manipulate the first K components, as the other components are ignored by the classifier. If adversarial examples rely on the last principal components (as Hendrycks *et. al* hypothesize), 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.

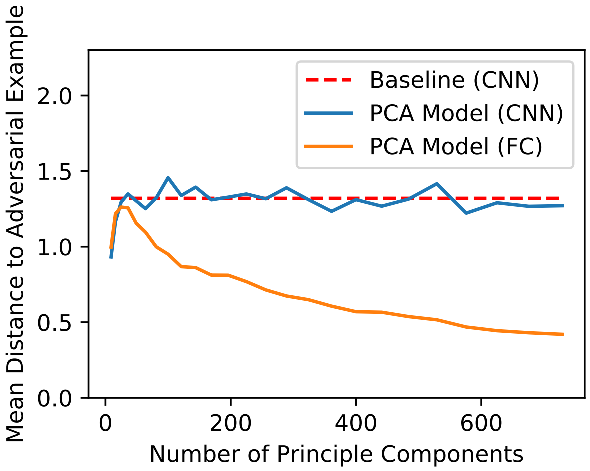

Figure 3: Performing dimensionality reduction increases the robustness of a 100-100-10 fully-connected neural network, but is still less secure than just using an unsecured CNN (the baseline). Dimensionality reduction does not help on a network that is already convolutional.

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

White-box Attack Evaluation. We evaluate Bhagoji's defense by constructing untargeted 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 feeding the first K principal components directly into 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 significantly 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.

5.3 Hidden Layer PCA

Li *et al.* [23] 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 valid only if all classifiers C_i accept the input, but rejects it if any do. Each classifier C_i is implemented as 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; based on experience with other defenses, we expect that attacking an ImageNet classifier would be even easier.

Oblivious Attack Evaluation. Li *et al.* generated adversarial examples with Szegedy *et. al*'s L-BFGS algorithm [36] 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). When we cascade this classifier, over 92% of valid instances are rejected as adversarial on MNIST and only 8% of adversarial examples are detected; for CIFAR, 98% of valid images are rejected and only 1% of adversarial examples are detected. This is even worse than random guessing. We conclude that Li's defense is not effective against the C&W attack.

6 DISTRIBUTIONAL DETECTION

Next, we study two defenses that detect adversarial examples by comparing the distribution of valid images to the distribution of adversarial examples. They use classical statistical methods to distinguish valid images from adversarial images. We show that neither of these defenses are able to detect a white-box or black-box adversary.

6.1 Maximum Mean Discrepancy

Grosse *et al.* [12] consider a significantly more powerful (perhaps so powerful as to be impractical) threat model: assume we are given two sets of images S_1 and S_2, such that we know S_1 contains only valid images, and we know that S_2 contains either all adversarial examples, or all valid 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 [4, 11], a statistical hypothesis test that answers the question "are these two sets drawn from the same underlying distribution?"

Let $\mathcal{F}$ represents a (possibly unbounded) set of functions. The optimal formulation of the MMD is

$$MMD(\mathcal{F}, X_1, X_2) = \sup_{f \in \mathcal{F}} E(f(X_1)) - E(f(X_2)).$$

It can be proven that if the support $|X_1|$ and $|X_2|$ are sufficiently large then the two distributions X_1, X_2 are equal if and only if the MMD is equal to zero.

Since this formal definition of MMD may not be computable, instead an approximation is used. We omit the exact details of the approximation used. In our experiments, we use the same approximation used by Grosse *et al.* [11].

To test whether X_1 and X_2 are drawn from the same distribution, Grosse *et al.* use a permutation test with the MMD test statistic:

1. Let $a = MMD_{\text{approx}}(X_1, X_2)$.
2. Let $Y = X_1 \cup X_2$, the set of all samples.
3. Randomly split Y into two halves, Y_1 and Y_2.
4. Test whether $a < MMD_{\text{approx}}(Y_1, Y_2)$.

This test is repeated many times (1000 in our implementation). If samples from the two distributions X_1 and X_2 are drawn from the same distribution, then we will fail to reject the null hypothesis with high probability, because the sets Y_1 and Y_2 will also be drawn from the same distribution. On the other hand, if X_1 and X_2 are drawn from different distributions, we should expect to reject the null hypothesis since Y_1 and Y_2 *are* drawn from the same distribution and so the MMD between them should be small.

To reproduce Grosse's results, we verify that the MMD test is able to distinguish weak attacks. As a first step, we verified that MMD correctly detects valid instances as being valid. To do this, we let the distribution X_1 be 1000 randomly sampled training instances, and X_2 be 1000 randomly sampled testing instances. MMD correctly fails to reject the null hypothesis ($p > 0.05$) in distinguishing the two distributions.

Then, we run this same experiment comparing random samples from the training set with on adversarial examples generated on the test set with weak attacks. Using a set size of 50, MMD can detect adversarial examples generated with FGS ($\epsilon = 0.275$) with $p < 0.001$ and detect JSMA adversarial examples (changing 20 pixels) with $p < 0.001$. This is consistent with the results reported by Grosse *et al.*

Oblivious Attack Evaluation. We repeated this experiment, this time producing untargeted 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). In short, Grosse's defense is not effective against oblivious attacks.

This should not be surprising: the MMD test acts directly on the pixel values. As long as the attack makes only slight changes to each image, it is unlikely a simple statistical test will be able to distinguish between adversarial examples and valid data.

We draw two important conclusions: first, MNIST is easier to protect than CIFAR (by a large margin), and second, we take this result as the strongest indication yet that FGS and JSMA should not be used exclusively as an evaluation benchmark.

6.2 Kernel Density Estimation

Feinman *et al.* [8] 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 valid images.

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

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

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 valid.

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.

Oblivious Attack Evaluation. Like many of the other defenses, we notice a difference in the efficacy of density estimation on the MNIST and CIFAR datasets. However, the difference is much more dramatic in this case. On MNIST, Feinman's defense can detect even adversarial examples generated using C&W's attack.

However, on CIFAR, simply using C&W's attack is enough to defeat density estimation. 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 (Figure 4, appendix). Therefore, Feinman's defense cannot work on CIFAR. In the remainder of this section, we show how to break this defense on MNIST.

White-box 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:

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

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 $\ell_2(x') \le 0$ 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 6 in the appendix shows adversarial examples generated against this defense.

Black-Box Evaluation. Note that our white-box attack algorithm doesn'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.

7 NORMALIZATION DETECTION

7.1 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 valid 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.

They use *dropout*, a common approach used during training to reduce network overfitting [34], for randomization. Dropout is applied to a layer of a network, and randomly sets a neuron's output to 0 with some probability p. Conventionally dropout is used only during training and is turned off during test-time, however Feinman *et al.* keep dropout turned on for both training and test. They use $p = 0.5$.

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 valid, 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 "valid" 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 $\tau = 0.02$.

Oblivious Attack Evaluation. We generated high-confidence adversarial examples with C&W's attack. 75% of these adversarial examples are detected, in contrast to 96% of examples generated with weaker attacks.

White-box Attack Evaluation. Our white-box attacks work by defining an objective function that combines the model and detector. We begin by showing a simple approach that was not effective at producing adversarial examples, to illustrate the challenges in selecting a useful function. Following the approach taken to break the prior detection schemes, let

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

(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 valid) then

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

We attempted to generated adversarial examples by applying C&W's attack to $G(\cdot)$. However, this failed: because values of the function $U(\cdot)$ are computed by calling $F(\cdot)$ (and not $Z(\cdot)$, as we like to do), 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 successfully generates adversarial examples that fool the dropout defense, with 98% success. The mean l_2 distortion is 3.68, up from the baseline of 2.05, so defeating the dropout defense requires the largest distortion of any of the defenses we have evaluated. Nonetheless, in an absolute sense, the adversarial examples are often indistinguishable from the originals, especially for the CIFAR dataset — see Figure 7. We show a plot of the uncertainty of valid images compared to adversarial images when constructed with our white-box attack in Figure 5 (appendix).

Black-Box Evaluation. It turns out that adversarial examples generated with the previous approach transfer with high accuracy 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.

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. We found 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.

7.2 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.

Oblivious 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×.

White-box 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.

8 LESSONS

After examining these ten defenses, we believe we have learned some general lessons about what worked, what didn't work, and advice on how to evaluate future defenses.

8.1 What Worked?

Applying randomness to the network (through dropout) was the most effective defense to our attacks on MNIST: it makes generating adversarial examples on the network as difficult as generating transferable adversarial examples. If it were possible to find a way to eliminate transferability, a randomization-based defense may be able to detect adversarial examples.

Kernel density estimation, the other defense that significantly increased the required distortion, was only effective on MNIST. We believe understanding why this defense works so well on MNIST but not at all on CIFAR is an interesting direction of future work.

8.2 What Didn't Work?

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. None of these approaches add more than a 30% increase in robustness on MNIST (and much less against CIFAR) when the adversary was aware of the model, and black-box attacks are also possible and almost as effective as white-box attacks.

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.)

Finally, for all defenses we evaluate, the transferability property allowed us to break them even if an adversary was not aware of the model parameters. Constructing a secure defense will require eliminating transferability.

8.3 Recommendations for Defenses

We have several recommendations for how researchers proposing new defenses can better evaluate their proposals:

Evaluate using a strong attack. Evaluate proposed defenses using the strongest attacks known. *Do not use fast gradient sign or JSMA exclusively*: these are weak attacks, and even if a defense can stop them, it is not possible to know if the defense is truly effective, or if it is simply the attack that is failing. Fast gradient sign was never designed to produce high-quality attacks. It was designed as a demonstration that neural networks are highly linear. As this paper has clearly demonstrated by breaking ten detection methods, JSMA is easily detected while other strong iterative attack algorithms are not. Using these algorithms as a first test is reasonable, but not sufficient. New schemes should demonstrate that they can stop C&W's attack.

Demonstrate white-box attacks fail. It is not sufficient to show that a defense can detect adversarial examples: one must also show that an adversary who is 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 valid by the detector, and apply a strong iterative attack (e.g., C&W's attack) to this function.

Demonstrate black-box attacks fail. If the scheme does not stop white-box attacks, at minimum it needs to stop black-box attacks. To evaluate security in the black-box model, we recommend generating high-confidence adversarial examples and testing how well they transfer.

Report mean distortion of adversarial examples. Many defenses we evaluated reported the success probability at a single distortion d. This makes it harder to compare multiple schemes, when they use different values of d. While a single number may not fully capture the robustness of a network, reporting the mean distance to the nearest adversarial example is a better single metric than the success rate at an arbitrary distance.

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 valid 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 CIFAR. 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 [7]).

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.

9 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 will remain 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. They all fail even in a black-box setting where the adversary only knows the technique the defender is planning on using but does *not* know the specific model parameters being used.

By studying these ten defenses, we have drawn two lessons: adversarial examples are much more difficult to detect than previously recognized, and existing defenses lack thorough security evaluations. We hope that our work will help raise the bar for evaluation of proposed defenses and perhaps help others to construct more effective defenses. We believe that constructing defenses to adversarial examples is a critical challenge that must be overcome before these networks are used in potentially security-critical domains.

10 ACKNOWLEDGEMENTS

We would like to thank Kathrin Grosse, Fuxin Li, Reuben Feinman, Metzen Jan Hendrik for discussing their defenses with us.

REFERENCES

[1] 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.

[2] 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).

[3] 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).

[4] 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.

[5] Katherine Bourzac. 2016. Bringing Big Neural Networks to Self-Driving Cars, Smartphones, and Drones. http://spectrum.ieee.org/computing/embedded-systems/bringing-big-neural-networks-to-selfdriving-cars-smartphones-and-drones. (2016).

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

[7] 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.

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

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

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

[11] 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.

[12] 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).

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

[14] 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.

[15] Jan Hendrik Metzen, Tim Genewein, Volker Fischer, and Bastian Bischoff. 2017. On Detecting Adversarial Perturbations. *arXiv preprint arXiv:1702.04267* (2017).

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

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

[18] Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167* (2015).

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

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

[21] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. 2016. Adversarial examples in the physical world. In *International Conference on Learning Representations (Workshop Track)*.

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

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

[24] 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.

[25] 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.

[26] 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).

[27] 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.

[28] 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).

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

[30] 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.

[31] 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).

[32] 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.

[33] 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)*.

[34] 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.

[35] 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.

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

[37] 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.

A ADDITIONAL FIGURES

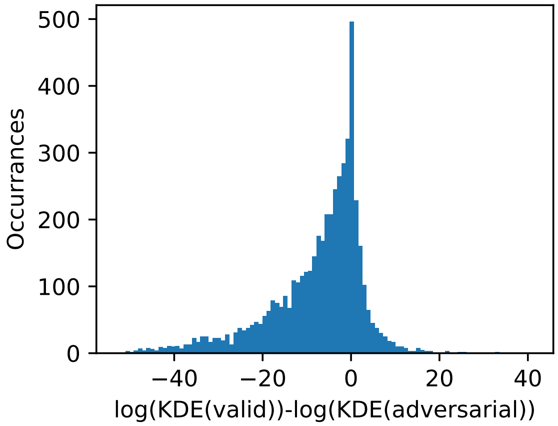

Figure 4: Density estimation fails to detect adversarial examples generated on the CIFAR dataset using a strong attack algorithm: 80% **of the time, the generated adversarial example is classified as *more likely* to be valid than the original example.**

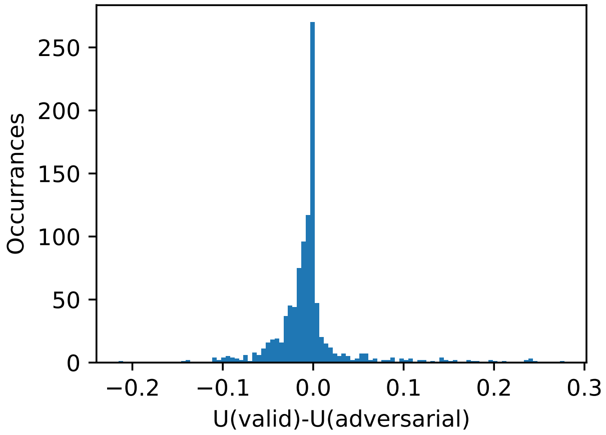

Figure 5: Dropout-based randomization does not detect adversarial examples generated with a white-box attack. The adversarial examples on average have lower uncertainty than the original valid image.

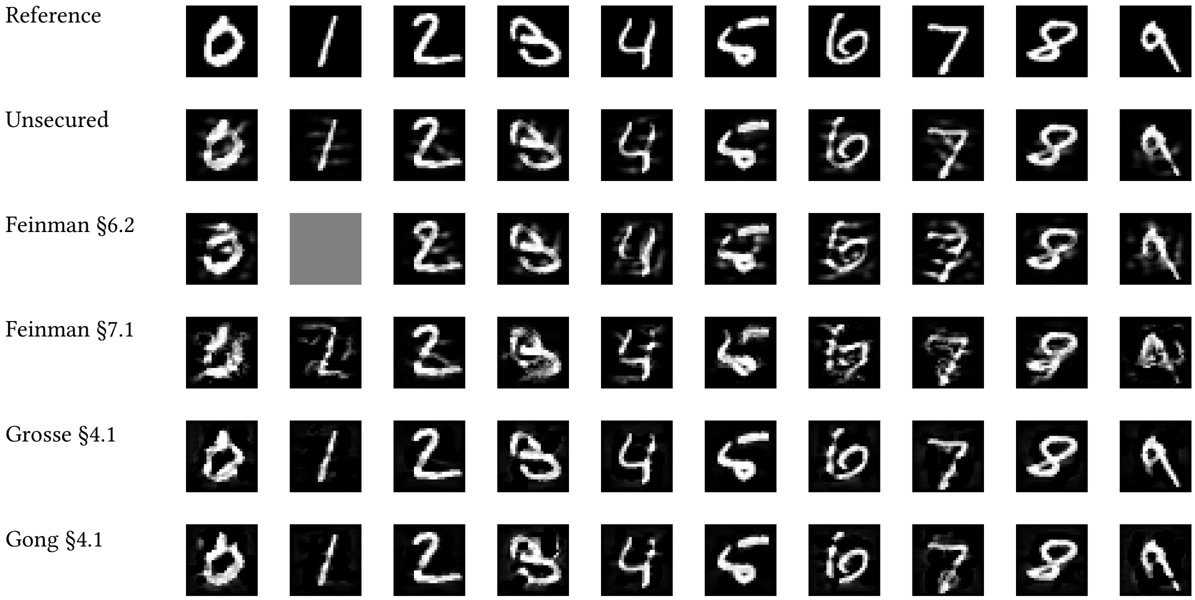

Figure 6: Adversarial examples on MNIST dataset. Each row corresponds to a defense where we construct an adaptive white-box attack. Defenses that were broken with an oblivious attack are omitted. Reference corresponds to the original (unmodified) image, unsecured to a baseline unsecured model, and the remaining to the defenses we study. The one grey image is caused by the attack failing to generate an adversarial example.

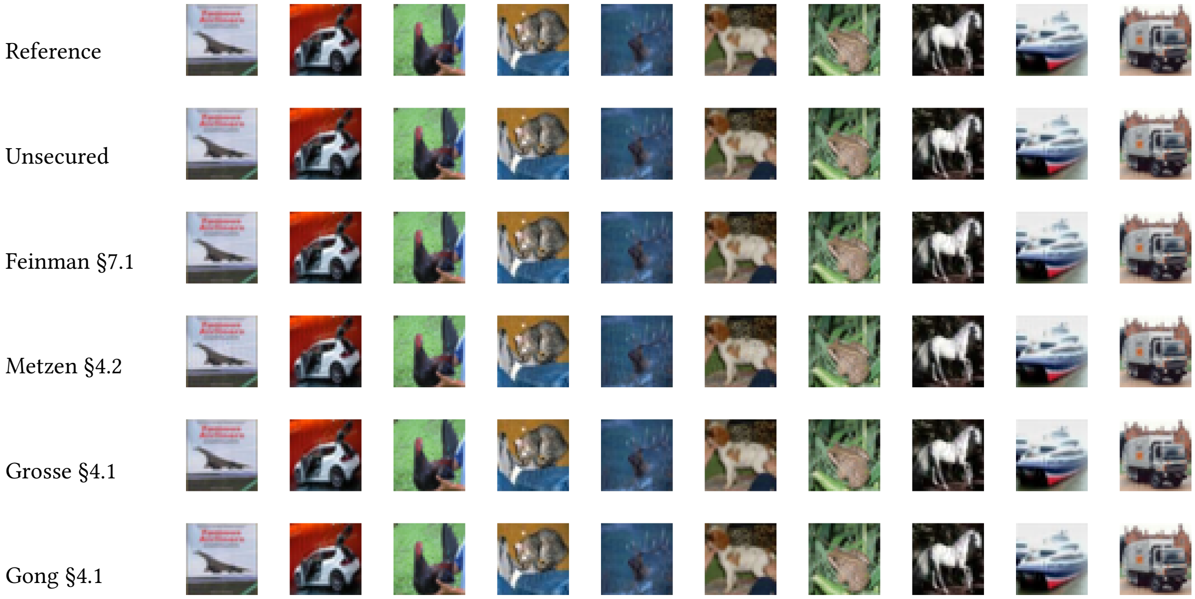

Figure 7: Adversarial examples on CIFAR dataset. Each row corresponds to a defense where we construct an adaptive white-box attack. Defenses that were broken with an oblivious attack are omitted. Reference corresponds to the original (unmodified) image, unsecured to a baseline unsecured model, and the remaining to the defenses we study.