Obfuscated Gradients Give a False Sense of Security: Circumventing Defenses to Adversarial Examples

Anish Athalye [*1] **Nicholas Carlini** [*2] **David Wagner** [2]

Abstract

We identify *obfuscated gradients* as a phenomenon that leads to a false sense of security in defenses against adversarial examples. While defenses that cause obfuscated gradients appear to defeat optimization-based attacks, we find defenses relying on this effect can be circumvented.

For each of the three types of obfuscated gradients we discover, we describe indicators of defenses exhibiting this effect and develop attack techniques to overcome it. In a case study, examining all defenses accepted to ICLR 2018, we find obfuscated gradients are a common occurrence, with 7 of 8 defenses relying on obfuscated gradients. Using our new attack techniques, we successfully circumvent all 7 of them.

1. Introduction

In response to the susceptibility of neural networks to adversarial examples (Szegedy et al., 2013), there has been significant interest recently in constructing defenses to increase the robustness of neural networks. While progress has been made in understanding and defending against adversarial examples, a complete solution has not yet been found. To the best of our knowledge, all defenses against adversarial examples published in peer-reviewed venues to date (Papernot et al., 2016; Hendrik Metzen et al., 2017; Hendrycks & Gimpel, 2017; Meng & Chen, 2017; Zantedeschi et al., 2017) are vulnerable to powerful optimization-based attacks (Carlini & Wagner, 2017c;b;a).

As benchmarking against iterative optimization-based attacks such as BIM (Kurakin et al., 2016a), PGD (Madry et al., 2018), and Carlini and Wagner's attack (Carlini & Wagner, 2017c) has become standard practice in evaluating potential defenses, new defenses have arisen that appear to withstand the most powerful optimization-based attacks.

We identify one common reason why many defenses provide apparent robustness against iterative attacks: *obfuscated gradients*. Without a good gradient signal, optimization-based methods cannot succeed. We identify three types of obfuscated gradients. Some defenses cause *shattered gradients*, either intentionally through non-differentiable operations or unintentionally through numerical instability, resulting in a nonexistent or incorrect gradient signal. Some defenses are randomized, causing *stochastic gradients* that depend on test-time entropy unavailable to the attacker. Other defenses cause *vanishing/exploding gradients* (Bengio et al., 1994), resulting in an unusable gradient signal.

We propose new techniques to overcome obfuscated gradients caused by these three phenomenon. We address gradient shattering due to non-differentiable operations with a new attack technique we call Backward Pass Differentiable Approximation. We compute gradients of randomized defenses by applying Expectation Over Transformation (Athalye et al., 2017). We solve vanishing/exploding gradients through reparameterization and optimize over a space where vanishing/exploding gradients are not an issue.

To investigate the prevalence of obfuscated gradients and understand the applicability of these attack techniques, we use the ICLR 2018 defenses as a case study. We find that obfuscated gradients are a common occurrence, with *7 of 8* accepted defenses relying on this phenomenon. Applying the new attack techniques we develop, we overcome obfuscated gradients and successfully circumvent all 7 of them. Along with this, we offer an analysis of the evaluations performed in the papers.

Additionally, we hope to provide researchers with a common baseline of knowledge, description of attack techniques, and common evaluation pitfalls, so that future defenses can avoid falling vulnerable to these same attacks.

To promote reproducible research, we release our reimplementation of each of these defenses, along with implementations of our attacks for each. [1]

[*]Equal contribution [1]Massachusetts Institute of Technology [2]University of California, Berkeley. Correspondence to: Anish Athalye <aathalye@mit.edu>, Nicholas Carlini <npc@berkeley.edu>.

[1] https://github.com/anishathalye/obfuscated-gradients

2. Preliminaries

2.1. Notation

We consider a neural network $f(\cdot)$ used for classification where $f(x)_i$ represents the probability image x corresponds to label i. We classify images, represented as $x \in [0,1]^{w \cdot h \cdot 3}$ for a 3-color image of width w and height h. We use $f^j(\cdot)$ to refer to layer j of the neural network, and $f^{1..j}(\cdot)$ the application of layers 1 through j. We denote the classification of the network as $c(x) = \arg\max_i f(x)_i$ where the true label of image x is written $c^*(x)$.

2.2. Adversarial Examples

Given an image x and classifier $f(\cdot)$, an adversarial example (Szegedy et al., 2013) x' satisfies two properties: $\mathcal{D}(x, x')$ is small for some distance metric $\mathcal{D}$, and $c(x') \neq c^*(x)$. That is, for images, x and x' appear visually similar but x' is classified incorrectly.

For this paper we use the ℓ_∞ and ℓ_2 distortion metrics to measure similarity. Two images which have a small distortion under either of these metrics will appear visually identical. We report ℓ_∞ distance in the normalized $[0,1]$ space, so that a distortion of 0.031 corresponds to $8/256$, and ℓ_2 distance as the total root-mean-square distortion normalized by the total number of pixels.

2.3. Datasets & Models

We evaluate these defenses on the same datasets on which they claim robustness: MNIST (LeCun, 1998) for Samangouei et al. (2018), CIFAR-10 (Krizhevsky & Hinton, 2009) for Madry et al. (2018); Song et al. (2018); Ma et al. (2018); Buckman et al. (2018); Dhillon et al. (2018), and ImageNet (Krizhevsky et al., 2012) for Guo et al. (2018); Xie et al. (2018). If a defense argues security on MNIST and any other dataset, we only circumvent the defense on the larger dataset. On MNIST and CIFAR-10, we evaluate defenses over the entire test set and generate untargeted adversarial examples. On ImageNet, we evaluate over 1000 randomly selected images in the test set, and construct *targeted* adversarial examples with randomly selected target classes. [2] Generating targeted adversarial examples is a strictly harder problem which we believe is also a more meaningful metric, especially for this dataset.

We use standard models for each dataset. For MNIST we use a standard convolutional neural network which reaches 99.3% accuracy. On CIFAR-10 we train a wide resnet (Zagoruyko & Komodakis, 2016; He et al., 2016) to 95% accuracy. For ImageNet we use the InceptionV3 (Szegedy et al., 2016) network which reaches 78.0% top-1 and 93.9% top-5 accuracy.

[2] Misclassification is a less meaningful metric on this dataset, where a misclassification of closely related classes, e.g. a German shepherd classified as a Doberman, may not be meaningful.

2.4. Attack Methods

We construct adversarial examples with iterative optimization-based methods. At a high level, for a given instance x, optimization attacks attempt to search for a δ such that $c(x+\delta) \neq c^*(x)$ either minimizing $\|\delta\|$, or maximizing the loss on $f(x+\delta)$. To generate ℓ_∞ bounded adversarial examples we use Projected Gradient Descent (PGD); for ℓ_2, we use a Lagrangian relaxation: Carlini and Wagner's formulation (Carlini & Wagner, 2017c). The specific choice of optimizer (e.g., gradient descent or Adam) and regularization (e.g., ℓ_∞-regularized or ℓ_2-regularized) is far less important than using optimization-based methods (Madry et al., 2018).

3. Obfuscated Gradients

At a high level, a defense obfuscates gradients when traveling in the direction suggested by the gradient is not a useful direction to travel in to construct an adversarial example.

We discover three ways in which defenses cause obfuscated gradients. We briefly define and discuss each of them.

Shattered Gradients are caused when a defense is non-differentiable, introduces numeric instability, or otherwise causes the true gradient signal to be *incorrect*. Defenses that cause gradient shattering often do so unintentionally, by introducing operations that are differentiable but where the gradient does not point in the direction that actually maximizes classification loss.

Stochastic Gradients are caused by randomized defenses, where either the network itself is randomized or the input is randomized before being fed to the classifier. Evaluating the gradient multiple times gives different results each time. This can cause single-step methods as well as optimization methods using a single sample of the randomness to incorrectly estimate the true gradient direction and fail to converge to a minima of the randomized classifier.

Exploding & Vanishing Gradients are often caused by defenses that consist of multiple iterations of neural network evaluation. Because this can be viewed as an extremely deep neural network evaluation, it is easy to see why gradients can either vanish or explode.

3.1. Identifying Obfuscated Gradients

In some cases it may be obvious that a defense contains a non-differentiable operation. However, in other cases, it may not be immediately clear. We discuss below several symptoms and diagnostic tests to help detect obfuscated gradients. While we do not claim these tests will perfectly

cover all cases, we find that every defense we are aware of that obfuscates gradients fails at least one of these tests.

Check that iterative attacks are better than single step. Iterative optimization-based attacks applied in a white-box setting are strictly stronger than single-step attacks and should give strictly superior performance. If single-step methods give performance superior to multi-step methods, it is likely that the multi-step attack is becoming stuck in its optimization search at a local minimum.

Verify white-box attacks work better than black-box. The black-box threat model is a strict subset of the white-box threat model, and so attacks in the white-box setting should perform better; however, if a defense is obfuscating gradients, then often times black-box attacks which do not require gradient signals will perform better than white-box attacks.

Ensure unbounded attacks reach 100% success. With unbounded distortion, any classifier should have 0% robustness to adversarial examples (as long as the classifier is not a constant function). If attacks do not reach 100% success, this indicates the defense is defeating the attack in a subtle manner and may not actually be increasing robustness.

Perform brute-force random sampling. A final simple method to identify obfuscated gradients is to perform brute-force search for adversarial examples (10^5 or more) within some ϵ-ball of each clean image. If random search finds adversarial examples not found with optimization methods, the defense is likely to be obfuscating gradients.

4. Attack Techniques

Generating adversarial examples through optimization-based methods requires useful gradients obtained through backpropagation (Rumelhart et al., 1986). Many defenses therefore either intentionally or unintentionally cause gradient descent to fail because of obfuscated gradients caused by gradient shattering, stochastic gradients, or vanishing/exploding gradients. We discuss a number of techniques that we develop to overcome obfuscated gradients.

4.1. Backward Pass Differentiable Approximation

Shattered gradients can be caused either unintentionally, e.g. by numerical instability, or intentionally, e.g. by using non-differentiable operations. To attack defenses where gradients are not readily available, we introduce a technique we call Backward Pass Differentiable Approximation (BPDA). [3]

[3]The BPDA approach can be used on an arbitrary network, even if it is already differentable, to obtain a more useful gradient.

4.1.1. Special Case

Many non-differentiable defenses can be expressed as follows: given a pre-trained classifier $f(\cdot)$, construct a preprocessor $g(\cdot)$ and let the secured classifier $\hat{f}(x) = f(g(x))$ where the preprocessor $g(\cdot)$ satisfies $g(x) \approx x$ (e.g., such a $g(\cdot)$ may perform image denoising to remove the adversarial perturbation). If $g(\cdot)$ is smooth and differentiable, then computing gradients through the combined network $\hat{f}$ is often sufficient to circumvent the defense (Carlini & Wagner, 2017b). However, recent work has constructed various functions $g(\cdot)$ which are neither smooth nor differentiable, and therefore can not be backpropigated through to generate adversarial examples with a white-box attack.

We introduce a new attack that we call Backward Pass Differentiable Approximation. Because g is constructed with the property that $g(x) \approx x$, we can approximate its derivative as the derivative of the identity function: $\nabla_x g(x) \approx \nabla_x x = 1$. Therefore, we can approximate the derivative of $f(g(x))$ at the point $\hat{x}$ as:

$$\nabla_x f(g(x))|_{x=\hat{x}} \approx \nabla_x f(x)|_{x=g(\hat{x})}$$

This allows us to compute gradients and therefore mount a white-box optimization attack. Conceptually, this attack is simple. We perform forward propagation through the neural network as usual, but on the backward pass, we replace $g(\cdot)$ with the identity function. In practice, the implementation can be expressed in an even simpler way: we approximate $\nabla_x f(g(x))$ by evaluating $\nabla_x f(x)$ at the point $g(x)$. This gives us an approximation of the true gradient, and while not perfect, is sufficiently useful that when averaged over many iterations of gradient descent still generates an adversarial example.

4.1.2. Generalized attack

While the above attack is effective for a simple class of networks expressable as $f(g(x))$ when $g(x) \approx x$, it is not fully general. We now generalize the above approach. Let $h(\cdot)$ be a smooth, differentable function, so that $f(x) \approx h(x)$.

To approximate $\nabla_x f(x)$ perform the forward pass $y = f(x)$ as is done typically. However, to perform the backward pass, backpropagate the value y through the function $h(x)$. As long as the two functions are similar, we find that the slightly inaccurate gradients still prove useful in constructing an adversarial example.

In practice, we do not use the completely general construction above. Instead, for functions of the form $f(g(x))$, where $g(x) \approx h(x)$ and $h(x)$ is differentiable, we approximate $\nabla_x f(g(x))$ by replacing $g(\cdot)$ with $h(\cdot)$ on the backward pass.

We have found applying BPDA is often necessary: replacing

$g(\cdot)$ with $h(\cdot)$ on both passes either is completely ineffective (e.g., with Song et al. (2018)) or many times less effective (e.g. with Buckman et al. (2018)).

4.2. Differentiating over Randomness

Stochastic gradients arise when using randomized transformations to the input before feeding it to the classifier or when using a stochastic classifier. When using optimization-based attacks on defenses that employ these techniques, it is necessary to estimate the gradient of the stochastic function.

Expectation over Transformation. For defenses that employ randomized transformations to the input, we apply Expectation over Transformation (EOT) (Athalye et al., 2017) to correctly compute the gradient over the expected transformation to the input.

When attacking a classifier $f(\cdot)$ that first randomly transforms its input according to a function $t(\cdot)$ sampled from a distribution of transformations T, EOT proposes optimizing the expectation over the transformation $\mathbb{E}_{t\sim T} f(t(x))$. The optimization problem can be solved by gradient descent, noting that $\nabla \mathbb{E}_{t\sim T} f(t(x)) = \mathbb{E}_{t\sim T} \nabla f(t(x))$, differentiating through the classifier and transformation, and approximating the expectation with samples at each gradient descent step.

Stochastic classifiers. For defenses that use stochastic classifiers, we correctly compute the gradient by computing gradients over the expectation of random parameters.

4.3. Reparameterization

We solve vanishing/exploding gradients by reparameterization. Assume we are given a classifier $f(g(x))$ where $g(\cdot)$ performs some optimization loop to transform the input x to a new input $\hat{x}$. Often times, this optimization loop means that differentiating through $g(\cdot)$, while possible, yields exploding or vanishing gradients.

To resolve this, we make a change-of-variable $x = h(z)$ for some function $h(\cdot)$ such that $g(h(z)) = h(z)$ for all z, but $h(\cdot)$ is differentable. For example, if $g(\cdot)$ projects samples to some manifold in a specific manner, we might construct $h(z)$ to return points exclusively on the manifold. This allows us to compute gradients through $f(h(z))$ and thereby circumvent the defense.

5. Case Study: ICLR 2018 Defenses

As a case study for evaluating the prevalence of obfuscated gradients, we study the ICLR 2018 defenses that argue robustness in a white-box threat model. We find that all but one of these defenses relies on this phenomenon to argue security, and we demonstrate that our techniques can circumvent each of those that rely on obfuscated gradients. We omit two defenses with provable security claims and one that only argues black-box security. We include one paper, Ma et al. (2018), that was not proposed as a defense *per se*, but suggests a method to detect adversarial exampls.

Defense	Dataset	Distance	Accuracy
Buckman et al. (2018)	CIFAR	0.031 (ℓ_∞)	0%∗
Ma et al. (2018)	CIFAR	0.031 (ℓ_∞)	5%
Guo et al. (2018)	ImageNet	0.007 (ℓ_2)	1%∗
Dhillon et al. (2018)	CIFAR	0.031 (ℓ_∞)	0%
Xie et al. (2018)	ImageNet	0.031 (ℓ_∞)	0%∗
Song et al. (2018)	CIFAR	0.031 (ℓ_∞)	9%∗
Samangouei et al. (2018)	MNIST	0.005 (ℓ_2)	0%
Madry et al. (2018)	CIFAR	0.031 (ℓ_∞)	47%

Table 1. **Summary of Results:** Seven of eight defense techniques accepted to ICLR 2018 cause obfuscated gradients and are vulnerable to our attacks. (Defenses denoted with ∗ also propose combining adversarial training; we report here the defense alone, see §5 for full numbers.)

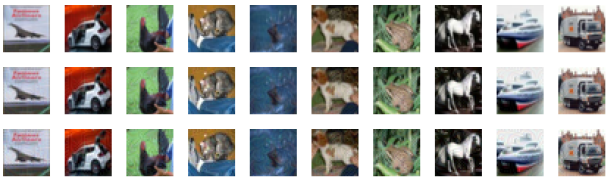

Figure 1. **Illustration of different distortion levels.** Row 1: Clean images. Row 2: Adversarial examples, distortion $\epsilon = 0.015$. Row 3: Adversarial examples, distortion $\epsilon = 0.031$.

There is an asymmetry in attacking defenses versus constructing robust defenses: to show a defense can be bypassed, it is only necessary to demonstrate one way to do so; in contrast, a defender must show no attack succeeds. We therefore only give one way to evade a defense when it can be done.

In Table 1 we summarize our results. Of the eight accepted papers, can be bypassed with our techniques because they rely on obfuscated gradients. Two of these defenses argue robustness on ImageNet, a much harder task than CIFAR-10; and one argues robustnes on MNIST, a much easier task than CIFAR-10. As such, comparing defenses across datasets is difficult.

Throughout this section we use a ℓ_∞ distortion of 0.015 and 0.031 on CIFAR-10 and ImageNet; we show in Figure 1 what such a distortion looks like for comparison.

5.1. A Secured Classifier

5.1.1. Adversarial Training

The first paper we consider (Madry et al., 2018) trains a high-capacity neural network to classify adversarial examples generated with optimization methods. We find that this approach does not cause gradient descent to fail in artificial ways.

Defense Details. Originally proposed by Szegedy et al. (2013), adversarial training is a conceptually simple process. Given training data $\mathcal{X}$ and loss function $\ell(\cdot)$, standard training chooses network weights θ as

$$\theta^* = \arg\min_\theta \ \mathbb{E}_{x\in\mathcal{X}} \ \ell(x; F_\theta).$$

Adversarial training instead chooses an ϵ-ball and solves the min-max formulation

$$\theta^* = \arg\min_\theta \ \mathbb{E}_{x\in\mathcal{X}} \left[\max_{\delta\in[-\epsilon,\epsilon]^N} \ell(x+\delta; F_\theta)\right].$$

To approximately solve this formulation, Madry et al. (2018) solve the inner maximization problem by generating adversarial examples the training data.

Discussion. The evaluation the authors perform for this defense pass all of our sanity checks. We find this approach does not cause obfuscated gradients and we are unable to substantially invalidate any of the claims made. However, we make two important observations about this defense: (1) we note that adversarial retraining has been shown to be difficult at ImageNet scale (Kurakin et al., 2016b); and (2) training exclusively on l_∞ adversarial examples provides only limited robustness to adversarial examples under other distortion metrics.

5.2. Gradient Shattering

5.2.1. Thermometer Encoding

Thermometer encoding (Buckman et al., 2018) is a encoding scheme designed to break the local linearity of neural networks. We find as a consequence it also causes gradient shattering and causes traditional optimization-based attacks to fail.

Defense Details. In contrast to prior work (Szegedy et al., 2013) which viewed adversarial examples as "blind spots" in neural networks, Goodfellow et al. (2014b) argue that the reason adversarial examples exist is that neural networks behave in a largely linear manner. The purpose of thermometer encoding is to break this linearity.

Given an image x, for each pixel color $x_{i,j,c}$, the l-level *thermometer encoding* $\tau(x_{i,j,c})$ is a l-dimensional vector where

$$\tau(x_{i,j,c})_k = \begin{cases} 1 & \text{if } x_{i,j,c} > k/l \\ 0 & \text{otherwise} \end{cases}.$$

For example, for a 10-level thermometer encoding, we have $\tau(0.66) = 1111110000$. The training process is identical between normal and thermometer networks.

Due to the discrete nature of thermometer encoded values, it is not possible to directly perform gradient descent on a thermometer encoded neural network. The authors therefore construct Logit-Space Projected Gradient Ascent (LS-PGA) as an attack over the discrete thermometer encoded inputs. Using this attack, the authors perform the adversarial training of Madry et al. (2018) on thermometer encoded networks.

On CIFAR-10, just performing thermometer encoding was found to give 50% accuracy within $\epsilon = 0.031$ under ℓ_∞ distortion. By performing adversarial training with 7 steps of LS-PGA, robustness increased to 80%.

Discussion. While the intention behind this defense is to break the local linearity of neural networks, we find that this defense in fact causes gradient shattering. This can be observed through their black-box attack evaluation: adversarial examples generated on a standard adversarially trained model transfer to a thermometer encoded model reducing the accuracy to 67%, well below the 80% robustness to the white-box iterative attack.

Evaluation. We use the BPDA approach from Section 4.1.2, where we let $g(x) = \tau(x)$. Observe that if we define

$$\hat{\tau}(x_{i,j,c})_k = \min(\max(x_{i,j,c} - k/l, 0), 1)$$

then

$$\tau(x_{i,j,c})_k = \text{floor}(\hat{\tau}(x_{i,j,c})_k)$$

so we can let $h(x) = \hat{\tau}(x)$ and replace the backwards pass with the function $h(\cdot)$.

LS-PGA reduces model accuracy to 50% on a thermometer-encoded model trained *without* adversarial training (bounded by $\epsilon = 0.031$). In constrast, we achieve 1% model accuracy with the lower $\epsilon = 0.015$ (and 0% with $\epsilon = 0.031$). This shows no measurable improvement from standard models, trained without thermometer encoding.

When we attack a thermometer-encoded adversarially trained model [4], we are able to reproduce the 80% accuracy at $\epsilon = 0.031$ claim against LS-PGA. However, our attack reduces model accuracy to 20%. This is significantly *weaker* than the 50% rate of success on the original Madry et al. (2018) model. Because this model is trained against

[4] That is, a thermometer encoded model that is trained using the approach of (Madry et al., 2018).

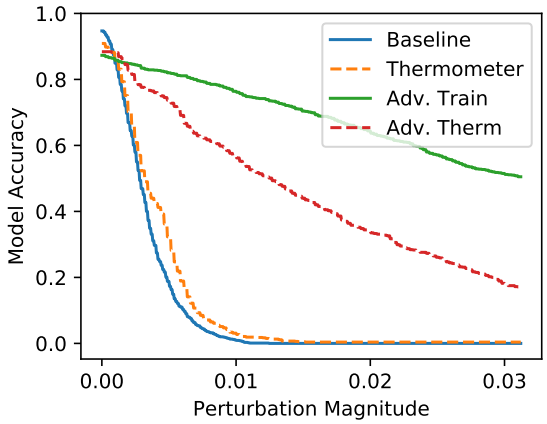

Figure 2. Model accuracy versus distortion (under l_∞). Adversarial training increases robustness to 50% at $\epsilon = 0.031$; thermometer encoding by itself provides limited value, and when coupled with adversarial training performs worse than adversarial training alone.

the (comparatively weak) LS-PGA attack, it is unable to adapt to the stronger attack we present above.

5.2.2. Local Intrinsic Dimensionality (LID)

The next paper studies properties of adversarial examples (Ma et al., 2018). They examine LID, a metric that measures the distance from an input compared to its neighbors, and suggest that LID might be useful for detecting adversarial examples. They present evidence that the LID is significantly larger for adversarial examples generated by existing attacks than for normal images, and they construct a classifier that can distinguish these adversarial images from normal images. The authors emphasize that this classifier *is not intended as a defense* against adversarial examples [5]. However, it would be natural to wonder whether it would be effective as a defense, so we study its robustness; our results confirm that it is not adequate as a defense. The method used to compute the LID relies on finding the k nearest neighbors, a non-differentiable operation, rendering gradient descent based methods ineffective.

Defense Details. The Local Intrinsic Dimensionality (Amsaleg et al., 2015) "assesses the space-filling capability of the region surrounding a reference example, based on the distance distribution of the example to its neighbors" (Ma et al., 2018). Let $\mathcal{S}$ be a mini-batch of N clean examples. Let $r_i(x)$ denote the distance (under metric $d(x, y)$) between sample x and its i-th nearest neighbor in $\mathcal{S}$ (under metric d). Then LID can be approximated by

$$\text{LID}_d(x) = -\left(\frac{1}{k}\sum_{i=1}^{k}\log\frac{r_i(x)}{r_k(x)}\right)^{-1}$$

where k is a defense hyperparameter the controls the number of nearest neighbors to consider. The authors use the distance function

$$d_j(x, y) = \left\|f^{1..j}(x) - f^{1..j}(y)\right\|_2$$

to measure the distance between the jth activation layers. The authors compute a vector of LID values for each sample:

$$\overrightarrow{\text{LID}}(x) = \{\text{LID}_{d_j}(x)\}_{j=1}^{n}.$$

Finally, they compute the $\overrightarrow{\text{LID}}(x)$ over the training data and adversarial examples generated on the training data, and train a logistic regression classifier to detect adversarial examples. We are grateful to the authors for releasing their complete source code.

Discussion. While LID is not a defense itself, the authors assess the ability of LID to detect different types of attacks.

Through solving the formulation

$$\text{min. } |x - x'|_2^2 + \alpha\left(\ell(x') + \text{LID-loss}(x')\right)$$

the authors attempt to determine if the LID metric is a good metric for detecting adversarial examples. Here, LID-loss$(\cdot)$ is a function that can be minimized to reduce the LID score. However, the authors report that this modified attack still achieves 0% success. Because Carlini and Wagner's ℓ_2 attack is unbounded, any time the attack does not reach 100% success indicates that the attack became stuck in a local minima. When this happens, it is often possible to slightly modify the loss function and return to 100% attack success (Carlini & Wagner, 2017b).

In this case, we observe the reason that performing this type of adaptive attack fails is that the LID computation is, while differentiable, not differentiable in a useful maner. Computing the LID term involves computing the k-nearest neighbors when computing $r_i(x)$. Minimizing the gradient of the distance to the current k-nearest neighbors is not representative of the true direction to travel in for the optimal set of k-nearest neighbors. As a consequence, we find that adversarial examples generated with gradient methods when penalizing for a high LID either (a) are not adversarial; or (b) are detected as adversarial, despite penalizing for the LID loss.

Evaluation. We now evaluate what would happen if a defense would directly apply LID to detect adversarial examples. Instead of performing gradient descent over a term that is difficult to differentiate through, we have found that generating high confidence adversarial examples (Carlini & Wagner, 2017a) (completely oblivious to to the detector) is sufficient to fool this detector. We obtain from the authors their detector trained on both the Carlini and Wagner's ℓ_2

[5] Personal communication with authors.

attack and train our own on the Fast Gradient Sign attack, both of which were found to be effective at detecting adversarial examples generated by other methods. By generating high-confidence adversarial examples minimizing ℓ_∞ distortion, we are able to reduce model accuracy to 2% success within $\epsilon = 0.015$. LID reports these adversarial examples are benign at a 97% rate (unmodified test data is flagged as benign with a 98% rate).

This evaluation demonstrates that the LID metric can be circumvented, and future work should carefully evaluate if building a detector relying on LID is robust to adversarial examples explicitly targeting such a detector. This work also raises questions whether a large LID is a fundamental characteristic of all adversarial examples, or whether it is a by-product of certain attacks.

5.2.3. Input Transformations

Guo et al. (2018) defend against adversarial examples through input transformations. They explore a number of functions for modifying the input before it is fed to the classifier. We find each of these transformations is non-differentiable and causes gradient shattering.

Defense Details. Guo et al. (2018) propose five input transformations to counter adversarial examples:

- Perform random *image cropping and rescaling*, averaging over multiple runs.
- Quantize images with *bit-depth reduction.*
- Apply *JPEG compression* to remove perturbations.
- Randomly drop pixels, and restore them by performing *total variance minimization*.
- *Image quilting:* Reconstruct images by replacing all 5×5 patches with patches from "clean" images, using minimum graph cuts in overlapping boundary regions to remove edge artifacts.

The authors explore different combinations of input transformations along with different underlying ImageNet classifiers, including adversarially trained models. They find that input transformations provide protection even with a vanilla classifier, providing varying degrees of robustness for varying transformations and normalized ℓ_2 perturbation budgets.

Discussion. The authors find that a ResNet-50 classifier provides a varying degree of accuracy for each of the five proposed input transformations [6] under the strongest attack with a normalized ℓ_2 dissimilarity of 0.01, with the strongest defenses achieving over 60% top-1 accuracy. We observe similar results when evaluating an InceptionV3 classifier.

The authors do not succeed in white-box attacks, crediting lack of access to test-time randomness as "particularly crucial in developing strong defenses" (Guo et al., 2018). [7]

Evaluation. It is possible to bypass each defense independently. We circumvent image cropping and rescaling with a direct application of Expectation Over Transformation (Athalye et al., 2017). To circumvent bit-depth reduction, JPEG compression, total variance minimization, and image quilting, we use BPDA to approximate the backward pass with the identity function. Using our attack, classification accuracy drops to 0.5% for the strongest defense under a perturbation budget 3 times smaller than the smallest perturbation budget considered in Guo et al. (2018), a root-mean-square perturbation of 0.007 (corresponding to a "normalized" ℓ_2 perturbation as defined in Guo et al. (2018) of 0.0003).

5.3. Stochastic Gradients

5.3.1. Stochastic Activation Pruning (SAP)

SAP randomly drops neurons at every layer of the network proportional to their absolute magnitude (Dhillon et al., 2018). By using sampling to compute gradients of the expectation over values of randomness, we generate adversarial examples that successfully attack this defense.

Defense Details. SAP randomly drops some neurons of each layer f^i to 0 according with probability proportional to their absolute value. For the hidden activation vector $h^i = f^{1..i}(x)$ at layer i, SAP defines a probability distribution

$$p^i_j = |h^i_j| \cdot \left(\sum_{k=1}^{m} h^i_m \right)^{-1}.$$

That is, p^i_j is proportional to the magnitude of h^i_j compared to the magnitude of the other neurons at this layer. SAP then computes a modified distribution

$$q^i_j = 1 - (1 - p^i_j)^r$$

where r is a defense hyperparameter (discussed below).

Then, each h^i_j is updated to a new value $\hat{h}^i_j$ by dropping it with probability q^i_j and keeping it otherwise

$$\hat{h}^i_j = \begin{cases} \frac{h^i_j}{q^i_j} & \text{with probability} \quad q^i_j \\ 0 & \text{with probability} \quad 1 - q^i_j \end{cases}$$

[6] The authors apply image cropping/rescaling, bit-depth reduction, and JPEG compression as baselines (Personal communication with authors).

[7] This defense may be stronger in a threat model where the adversary does not have complete information about the exact quilting process used (Personal communication with authors).

so that values less likely to be sampled are scaled up accordingly. The value r is chosen to keep a large enough fraction of the neurons that the accuracy remains high, but not so large that all neurons are kept. We follow the authors advice and choose r so the test accuracy drops by only 5%.

Discussion. The authors only evaluate SAP by taking a single step in the gradient direction (Dhillon et al., 2018). While taking a single step in the direction of the gradient is effective on non-randomized neural networks, when randomization is used, computing the gradient with respect to one sample of the randomness is ineffective.

Evaluation. To resolve this difficult, we estimate the gradients by computing the expectation over instantiations of randomness. At each iteration of gradient descent, instead of taking a step in the direction of $\nabla_x f(x)$ we move in the direction of $\sum_{i=1}^{k} \nabla_x f(x)$ where each invocation is randomized with SAP. We have found that choosing $k = 10$ provides useful gradients. We additionally had to resolve a numerical instability when computing gradients: this defense caused computing a backward pass to cause exploding gradients due to division by numbers very close to 0. We resolve this by clipping gradients or through stable numerical techniques.

With these approaches, we are able to reduce SAP model accuracy to 9% at $\epsilon = .015$, and 0% at $\epsilon = 0.031$. If we consider an attack successful only when an example is classified incorrectly 10 times out of 10 (and consider it correctly classified if it is ever classified as the correct label), model accuracy is below 10% with $\epsilon = 0.031$.

5.3.2. Mitigating through Randomization

Xie et al. (2018) defend against adversarial examples by randomly rescaling and padding images so that natural images retain their classification while adversarial examples are sufficiently perturbed to lose their adversarial nature. We circumvent this defense by using EOT to synthesize robust adversarial examples (Athalye et al., 2017).

Defense Details. (Xie et al., 2018) propose to defend against adversarial examples by adding a randomization layer before the input to the classifier. For a classifier that takes a 299×299 input, the defense first randomly rescales the image to a $r \times r$ image, with $r \in [299, 331)$, and then randomly zero-pads the image so that the result is 331×331. The output is then fed to the classifier.

Discussion. The authors consider three attack scenarios: vanilla attack (an attack on the original classifier), single-pattern attack (an attack assuming some fixed randomization pattern), and ensemble-pattern attack (an attack over a small ensemble of fixed randomization patterns). The authors strongest attack reduces InceptionV3 model accuracy to 32.8% top-1 accuracy (over images that were originally classified correctly).

The authors dismiss a stronger attack over larger choices of randomness, stating that it would be "computationally *impossible*" (emphasis ours) and that such an attack "may not even converge" (Xie et al., 2018).

Evaluation. We find the authors ensemble-pattern attack overfits to the ensemble with fixed randomization. We bypass this defense by applying Expectation over Transformation (Athalye et al., 2017), optimizing over the (in this case, discrete) distribution of transformations T and minimizing $\mathbb{E}_{t \sim T} f(t(x))$.

We approximate the gradient of the above by sampling and differentiating through the transformation. Using this attack, even if we consider the attack successful only when an example is classified incorrectly 10 times out of 10, we can reduce the accuracy of the classifier from 32.8% to 0.0% with a maximum ℓ_∞ perturbation of $\epsilon = 0.031$.

5.4. Vanishing & Exploding Gradients

5.4.1. PixelDefend

Song et al. (2018) propose using a PixelCNN generative model to project a potential adversarial example back onto the data manifold before feeding it into a classifier. We bypass this defense using BPDA.

Defense Details. Song et al. (2018) argue that adversarial examples mainly lie in the low-probability region of the training distribution. PixelDefend "purifies" adversarially perturbed images by projecting them back onto the data manifold through the use of a PixelCNN generative model, and then it feeds the resulting image through an unmodified classifier. PixelDefend uses a greedy decoding procedure to approximate finding the highest probability example within an ϵ-ball of the input image.

Discussion. The authors evaluate PixelDefend on CIFAR-10 over a number of underlying classifiers, perturbation budgets, and attack algorithms.

With a maximum ℓ_∞ perturbation of $\epsilon = 0.031$ on CIFAR-10, PixelDefend claims 46% accuracy (with a vanilla ResNet classifier). The authors dismiss the possibility of end-to-end attacks on PixelDefend due to the difficulty of differentiating through an unrolled version of PixelDefend due to vanishing gradients and computation cost.

Evaluation. We sidestep the problem of computing gradients through an unrolled version of PixelDefend by approximating gradients with BPDA, approximating the backward

pass with the identity function, and we successfully mount an end-to-end attack using this technique [8]. With this attack, we can reduce the accuracy of a naturally trained classifier which achieves 95% accuracy to 9% with a maximum ℓ_∞ perturbation of $\epsilon = 0.031$. Using the adversarially trained classifier of Madry et al. (2018), using PixelDefend provides no additional robustness over just using the adversarially trained classifier.

5.4.2. Defense-GAN

Defense-GAN (Samangouei et al., 2018) uses a Generative Adversarial Network (Goodfellow et al., 2014a) to project samples onto the manifold of the generator before classifying them. We use reparameterization to circumvent the multi-step projection process and construct adversarial examples.

Defense Details. The defender first trains a Generative Adversarial Network with a generator $G(z)$ that maps samples from a latent space (typically $z \sim \mathcal{N}(0, 1)$) to images that look like training data. Defense-GAN takes a trained classifier $f(\cdot)$, and to classify an input x, instead of returning $f(x)$, returns $f(\arg\min_z |G(z) - x|)$. To perform this projection to the manifold, the authors take many steps of gradient descent starting from different random initializations.

Defense-GAN was not shown to be effective on CIFAR-10. We therefore evaluate it on MNIST (where it was argued to be secure).

Discussion. In Samangouei et al. (2018), the authors construct a white-box attack by unrolling the gradient descent used during classification. Despite an unbounded ℓ_2 perturbation size, Carlini and Wagner's attack only reaches 30% misclassification rate on the most vulnerable modele and under 5% on the strongest. This leads us to believe that unrolling gradient descent breaks gradients.

Evaluation. Performing the manifold projection is non-trivial as an inner optimization step when generating adversarial examples. To sidestep this difficulty, we show that adversarial examples exist *directly on* the data manifold. That is, we construct an adversarial example $x' = G(z^*)$ so that $|x - x'|$ is small and $c(x) \neq c(x')$.

To do this, we solve the re-parameterized formulation

$$\text{min.} \quad \|G(z) - x\|_2^2 + c \cdot \ell(G(z)).$$

[8] In place of a PixelCNN, due to the availability of a pre-trained model, we use a PixelCNN++ (Salimans et al., 2017) and discretize the mixture of logistics to produce a 256-way softmax over possible pixel values. Due to compute limitations, we evaluate our attack over a random sample of 500 images from the test set.

Figure 3. Images on the MNIST test set. Row 1: Clean images. Row 2: Adversarial examples on an unsecured classifier. Row 3: Adversarial examples on Defense-GAN.

We initialize $z = \arg\min_z |G(z) - x|$ (also found via gradient descent). We train a WGAN using the code the authors provide (Gulrajani et al., 2017), and a MNIST CNN to 99.3% accuracy.

We run for 50k iterations of gradient descent for generating each adversarial example; this takes under one minute per instance. The unsecured classifier requires a mean ℓ_2 distortion of 0.0019 (per-pixel normalized, 1.45 un-normalized) to fool. When we mount our attack on Defense-GAN, we require a mean distortion of 0.0027, an increase in distortion of $1.46\times$; see Figure 3 for examples of adversarial examples. The reason our attacks succeed with 100% success without suffering from vanishing or exploding gradients is that our gradient computation only needs to differentiate through the generator $G(\cdot)$ once.

Concurrent to our work, Ilyas et al. (2017) also develop a nearly identical approach to Defense-GAN; they also find it is vulnerable to the attack we outline above, but increase the robustness further with adversarial training. We do not evaluate this extended approach.

6. Discussion

Having demonstrated attacks on these seven papers, we now take a step back and describe what we believe to be a complete method of evaluating a defense to adversarial examples. Much of the advice we give has been given in prior work (Carlini & Wagner, 2017a; Madry et al., 2018); we repeat it here and offer our own perspective for completeness. We hope future work can use this as a guide for performing an evaluation.

6.1. Define a (realistic) threat model

When constructing a defense, it is critical to define a threat model that limits the adversary. Prior work used words *white-box*, *grey-box*, *black-box*, *no-box* to describe slightly different threat models, often overloading the same word.

Instead of attempting to, yet again, redefine the vocabulary, we outline the various aspects of a defense that might be

revealed to the adversary or held secret to the defender:

- *Model architecture* and *Model weights*
- *Training algorithm* and *Training data*
- For defenses that involve *randomness*, whether the adversary knows the exact sequence of random values that will be chosen, or only the distribution.
- If assuming the adversary does not know the model architecture and weights, if *query access* is allowed. If so, if the model output is the logits, probability vector, or predicted label (i.e., arg max).

While there are some aspects of a defense that might be held secret, threat models should not contain unrealistic constraints. We believe any compelling threat model will at the very least grant knowledge of the model architecture, training algorithm, and allow query access.

We do not believe it is meaningful to restrict the the computational power of an adversary. If two defenses are equally robust but generating adversarial examples on one takes one second and another takes ten seconds, the robustness *has not increased*. Only if the adversary's computational effort can be shown to be increased exponentially faster than prediction runtime *might* it be it acceptable to use runtime as a security parameter. However, increasing attack time by a few seconds (as occurs in the defenses we attack in this paper) is not meaningful.

6.2. Make specific, testable claims

After defining a clear threat model, a defense should make make specific, testable claims. For example, a defense might claim 90% robustness to ℓ_∞ adversarial examples of distortion at most $\epsilon = 0.031$, or might claim that mean ℓ_2 distortion to adversarial examples increases by a factor of two from a baseline model to a secured model (in which case, the baseline should also be clearly defined).

Unfortunately, many of the defenses we evaluate simply claim they are robust without giving specific bounds. The biggest violation of this advice is that defense should never claim complete robustness against *unbounded* attacks: with unlimited distortion any image can be converted into any other, yielding 100% "success".

In order to allow the claims to be testable, the defense must be specified completely, with all hyperparameters given. Releasing source code and a pre-trained model along with the paper is perhaps the most useful method of making explicit claims. Four of the defenses we study made complete source code available (Madry et al., 2018; Ma et al., 2018; Guo et al., 2018; Xie et al., 2018).

6.3. Evaluate against adaptive attacks

Claiming increased robustness to existing attacks, while specific and testable, is not by itself a useful claim. It is important to actively evaluate one's own defense with new defense-aware attacks to justify claims of security.

In particular, once a defense has been completely specified, it is important to attempt to circumvent that concrete defense, assuming only that the adversary is restricted to the threat model. If it can be circumvented, then it is important to *not* give ways to prevent that specific attack (i.e., by tweaking a hyperparameter). After an evaluation, it is acceptable to modify the defense, but then a new attack should be built to target the newly modified defense. In this way, concluding an evaluation with a final adaptive attack can be seen as analogous to evaluating a model on the test data.

7. Conclusion

Constructing defenses to adversarial examples requires defending against not only existing attacks but also future attacks that may be developed. In this paper, we identify *obfuscated gradients*, a phenomenon exhibited by certain defenses that makes standard gradient-based methods fail to generate adversarial examples. We develop three attack techniques to bypass three different types of obfuscated gradients. To evaluate the applicability of our techniques, we use the ICLR 2018 defenses as a case study, circumventing seven of eight accepted defenses.

More generally, we hope that future work will be able to avoid relying on obfuscated gradients for perceived robustness and use our evaluation approach to detect when this occurs. Defending against adversarial examples is an important area of research and we believe performing a thorough evaluation is a critical step that can not be overlooked.

Acknowledgements

We are grateful to Aleksander Madry and Andrew Ilyas for helpful comments on an early draft of this paper. We thank Bo Li, Xingjun Ma, Laurens van der Maaten, Aurko Roy, Yang Song, and Cihang Xie for useful discussion and insights on their defenses.

This work was partially supported by the National Science Foundation through award CNS-1514457, Qualcomm, and the Hewlett Foundation through the Center for Long-Term Cybersecurity.

References

Amsaleg, Laurent, Chelly, Oussama, Furon, Teddy, Girard, Stéphane, Houle, Michael E, Kawarabayashi, Ken-ichi, and Nett, Michael. Estimating local intrinsic dimension-

ality. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 29–38. ACM, 2015.

Athalye, Anish, Engstrom, Logan, Ilyas, Andrew, and Kwok, Kevin. Synthesizing robust adversarial examples. *arXiv preprint arXiv:1707.07397*, 2017.

Bengio, Y., Simard, P., and Frasconi, P. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, Mar 1994. ISSN 1045-9227. doi: 10.1109/72.279181.

Buckman, Jacob, Roy, Aurko, Raffel, Colin, and Goodfellow, Ian. Thermometer encoding: One hot way to resist adversarial examples. *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=S18Su--CW. accepted as poster.

Carlini, Nicholas and Wagner, David. Adversarial examples are not easily detected: Bypassing ten detection methods. *AISec*, 2017a.

Carlini, Nicholas and Wagner, David. Magnet and "efficient defenses against adversarial attacks" are not robust to adversarial examples. *arXiv preprint arXiv:1711.08478*, 2017b.

Carlini, Nicholas and Wagner, David. Towards evaluating the robustness of neural networks. In *IEEE Symposium on Security & Privacy*, 2017c.

Dhillon, Guneet S., Azizzadenesheli, Kamyar, Bernstein, Jeremy D., Kossaifi, Jean, Khanna, Aran, Lipton, Zachary C., and Anandkumar, Animashree. Stochastic activation pruning for robust adversarial defense. *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=H1uR4GZRZ. accepted as poster.

Goodfellow, Ian, Pouget-Abadie, Jean, Mirza, Mehdi, Xu, Bing, Warde-Farley, David, Ozair, Sherjil, Courville, Aaron, and Bengio, Yoshua. Generative adversarial nets. In *Advances in neural information processing systems*, pp. 2672–2680, 2014a.

Goodfellow, Ian J, Shlens, Jonathon, and Szegedy, Christian. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014b.

Gulrajani, Ishaan, Ahmed, Faruk, Arjovsky, Martin, Dumoulin, Vincent, and Courville, Aaron. Improved training of wasserstein gans. *arXiv preprint arXiv:1704.00028*, 2017.

Guo, Chuan, Rana, Mayank, Cisse, Moustapha, and van der Maaten, Laurens. Countering adversarial images using input transformations. *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=SyJ7ClWCb. accepted as poster.

He, Kaiming, Zhang, Xiangyu, Ren, Shaoqing, and Sun, Jian. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Hendrik Metzen, Jan, Genewein, Tim, Fischer, Volker, and Bischoff, Bastian. On detecting adversarial perturbations. In *International Conference on Learning Representations*, 2017.

Hendrycks, Dan and Gimpel, Kevin. Early methods for detecting adversarial images. In *International Conference on Learning Representations (Workshop Track)*, 2017.

Ilyas, Andrew, Jalal, Ajil, Asteri, Eirini, Daskalakis, Constantinos, and Dimakis, Alexandros G. The robust manifold defense: Adversarial training using generative models. *arXiv preprint arXiv:1712.09196*, 2017.

Krizhevsky, Alex and Hinton, Geoffrey. Learning multiple layers of features from tiny images. 2009.

Krizhevsky, Alex, Sutskever, Ilya, and Hinton, Geoffrey E. ImageNet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105, 2012.

Kurakin, Alexey, Goodfellow, Ian, and Bengio, Samy. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533*, 2016a.

Kurakin, Alexey, Goodfellow, Ian J., and Bengio, Samy. Adversarial machine learning at scale. *arXiv preprint arXiv:1611.01236*, 2016b.

LeCun, Yann. The mnist database of handwritten digits. *http://yann.lecun.com/exdb/mnist/*, 1998.

Ma, Xingjun, Li, Bo, Wang, Yisen, Erfani, Sarah M., Wijewickrema, Sudanthi, Schoenebeck, Grant, Houle, Michael E., Song, Dawn, and Bailey, James. Characterizing adversarial subspaces using local intrinsic dimensionality. *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=B1gJ1L2aW. accepted as oral presentation.

Madry, Aleksander, Makelov, Aleksandar, Schmidt, Ludwig, Tsipras, Dimitris, and Vladu, Adrian. Towards deep learning models resistant to adversarial attacks. *International Conference on Learning Representations*,

2018. URL https://openreview.net/forum?id=rJzIBfZAb. accepted as poster.

Meng, Dongyu and Chen, Hao. MagNet: a two-pronged defense against adversarial examples. In *ACM Conference on Computer and Communications Security (CCS)*, 2017. arXiv preprint arXiv:1705.09064.

Papernot, Nicolas, McDaniel, Patrick, Wu, Xi, Jha, Somesh, and Swami, Ananthram. Distillation as a defense to adversarial perturbations against deep neural networks. In *Security and Privacy (SP), 2016 IEEE Symposium on*, pp. 582–597. IEEE, 2016.

Rumelhart, David E., Hinton, Geoffrey E., and Williams, Ronald J. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.

Salimans, Tim, Karpathy, Andrej, Chen, Xi, and Kingma, Diederik P. Pixelcnn++: A pixelcnn implementation with discretized logistic mixture likelihood and other modifications. In *ICLR*, 2017.

Samangouei, Pouya, Kabkab, Maya, and Chellappa, Rama. Defense-gan: Protecting classifiers against adversarial attacks using generative models. *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=BkJ3ibb0-. accepted as poster.

Song, Yang, Kim, Taesup, Nowozin, Sebastian, Ermon, Stefano, and Kushman, Nate. Pixeldefend: Leveraging generative models to understand and defend against adversarial examples. *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=rJUYGxbCW. accepted as poster.

Szegedy, Christian, Zaremba, Wojciech, Sutskever, Ilya, Bruna, Joan, Erhan, Dumitru, Goodfellow, Ian, and Fergus, Rob. Intriguing properties of neural networks. *ICLR*, 2013.

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

Xie, Cihang, Wang, Jianyu, Zhang, Zhishuai, Ren, Zhou, and Yuille, Alan. Mitigating adversarial effects through randomization. *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=Sk9yuql0Z. accepted as poster.

Zagoruyko, Sergey and Komodakis, Nikos. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

Zantedeschi, Valentina, Nicolae, Maria-Irina, and Rawat, Ambrish. Efficient defenses against adversarial attacks. *arXiv preprint arXiv:1707.06728*, 2017.