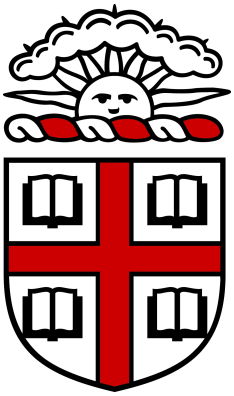
BROWN UNIVERSITY



# An Exposition of Adversarial Examples in Neural Networks

Di Yang (Steven) Shi

Submitted in partial fulfillment of the requirements for earning
Honors in Computer Science

Advisor: Vasileios Kemerlis
Reader: Thomas Serre

Department of Computer Science

April 2018

# *Acknowledgements*

Special thanks to Vasileios Kemerlis for his constant support in advising me, Thomas Serre for being a reader, and Nicholas Carlini (UC Berkeley) and Anish Athalye (MIT) for guiding me through some of their papers and the general space.

# Contents

# Chapter 1

# Introduction

Neural networks have become both a staple and a standard in the field of machine learning. From reinventing solutions to classic use cases such as audiovisual recognition [1–3], decision making in games [4, 5], and language processing [6, 7], to more contemporary use cases such as mimicking creativity in music [8], pictures [9], and paintings [10], neural networks have permeated everyday life at an astounding rate. We already rely on them in some security critical areas, such as detection and decision systems in self-driving cars, pattern detection authentication services, and parsing privileged commands from speech. As they continue integrating into society, the risks of leaving vulnerabilities in the technology grows exponentially. Thus, it is imperative that we thoroughly understand the vulnerabilities that adversaries can exploit and employ countermeasure defenses accordingly.

## 1.1 History

The first published mention of adversarial examples by Biggio *et al.* [11] introduces the concept of and mandates a strong consideration for the security of machine learning systems. In particular, Biggio *et al.* describes a so-called evasion attack, which can be formulated as non-linear (and generally non-convex) optimization

problem in both perfect knowledge and limited knowledge scenarios. An immediate followup by Szegedy *et al.* [12] detailed an evasion attack in the image classification domain. While their work focused mainly on the neural network space, the concepts are the same. By introducing minor perturbations undetectable by the human eye, their adversarial examples were able to fool then-state of the art image classifiers.

Perhaps it is a phenomenon of academics that a topic is introduced, mysteriously disappears for some number of years, and then resurges with great popularity. Such is the case with adversarial examples. In the previous year, attacks and defenses have been proposed step in step at such a pace that a defense [13] was published several months after an attack that already broke it [14]. As the synthesizing of 3-D adversarial examples has proven to be successful [14], the search for a robust defense against adversarial examples has become an increasingly important question.

## 1.2 Organization

We will begin by formally describing what neural networks are, along with their associated constructs. Then, we will introduce the notion of adversarial examples and notable attacks and defenses relating to them. From a security perspective, we will illuminate aspects of the threat model particular to neural networks. We then transition to a more applied setting of the recent state of adversarial examples and their prior barrier in being able to persist as a physical form in the real world. Finally, we attempt to extend the threat models to persist through real world perturbations, reproducing the state-of-the-art results [14, 15].

In this work, we focus primarily on classification networks in the computer vision space as this is the space where adversarial examples have been most researched.

# Chapter 2

# Background

## 2.1 Neural Networks

**Definition 2.1.** Given some $n$-dimensional input space $D \subseteq \mathbb{R}^d$ and a set of valid discrete labels $L = \{1, \ldots, k\}$, we define a **classifier** as $f : D \to L$. In short, for some input $x \in D$ and label $\ell \in L$, $f(x) = \ell$ is equivalent to saying that the classifier believes $l$ is the most-likely label of $x$. Typically, there is an intermediate step where a probability vector is produced which contains the likelihood of any given $\ell \in L$; these are called **soft labels**. Then, $f(x) = \arg\max_{\ell} P(\ell|x)$ is called a **hard label**.

In general, a distance function $d : D \times D \to \mathbb{R}_{\geq 0}$ is provided such that $(D, d)$ is a metric space.

**Definition 2.2.** Let $F$ be the underlying **ground truth** function we are trying to model with $f$. Then, $F(x) = y_{\text{true}}$ is equivalent to saying that $y_{\text{true}}$ is the **ground truth** label for $x$.

**Definition 2.3.** It is presumed that $f$ has an associated **loss function** $\text{loss}_f : D \times L \to \mathbb{R}_{\geq 0}$ which denotes a measure on the cost of incorrectness in our model. For classification problems, cross-entropy is a widely used choice.

**Definition 2.4.** The classification rate or **accuracy** of a classifier, $f$, on a distribution of inputs, $X$, is defined as

$$C_f(X) = \sum_{x \in X} \frac{\mathbb{1}_{\text{true}}(f(x) = F(x))}{|X|}$$

Then, the misclassification rate is simply $1 - C_f(X)$ or defined as

$$M_f(X) = \sum_{x \in X} \frac{\mathbb{1}_{\text{true}}(f(x) \neq F(x))}{|X|}$$

### 2.1.1 Layers

Neural networks are composed of the input layer, hidden layers (computational layers between input and output layers), and the output layer. Each layer is a collection of nodes which store computed values and the edges between nodes represent some nonlinear transformation, i.e. each hidden layer is some nonlinear transformation of the previous layer(s). For our purposes, we will briefly discuss the softmax layer.

Where $\langle x_i \rangle$ is some vector of values (presumably the final hidden layer before the softmax layer) and $\ell$ is some label, the softmax function $\sigma : \mathbb{R}^n \to [0, 1]^n$ is defined as

$$\sigma_x(x_\ell) = \frac{e^{x_\ell}}{\sum\limits_{x_\ell \in x} e^{x_\ell}}$$

In short, the softmax function normalizes the element-wise exponentiation of a vector of values and forces the sum of them to be one. The softmax layer is simply the softmax function applied to each element of the previous layer, which is a vector of values in $\mathbb{R}$ representing the confidence of each label. Thus, we treat the result of the softmax layer as the probabilities of the classifier even though the output is parameterized by the model itself.

### 2.1.2 Backpropagation

In discussing neural networks, the term "learning" typically refers to the training of parameters in fitting $f$ to $F$ on the training set and validation set. Specifically, the backpropagation algorithm is used for neural networks to update their parameters.

First, the "forward pass" consists of evaluating the network on the input, thus storing computed values in the nodes of the graph. Let $\theta$ denote the trainable parameters of our classifier, $f$, and $\eta$ be the learning rate. The backpropagation step is defined as follows

$$\theta_i = \theta_{i-1} - \eta \nabla_{\theta_{i-1}} loss_{f_\theta}(x, y_{\text{true}})$$

In other words, for our trainable parameters $\theta$, take a step of magnitude scaled with $\eta$ in the direction which minimizes the loss of classifying $x$ as $y_{\text{true}}$.

### 2.1.3 Assumptions

As many (if not all) of the attacks are gradient based, we merely require that the neural network is differentiable to launch an attack on it. However, it should be noted that the gradient does not even need to necessarily be usable, it just needs to exist. Such an assumption is fairly reasonable as neural networks widely use the backpropagation algorithm, which relies on the gradient existing through the network, to update its parameters. Moreover, it is assumed that we can query the classifier at some reasonable rate, i.e. the ability to sample the input and output pairs it is defined on.

## 2.2 Adversarial Examples

**Definition 2.5.** An **adversarial example** [11, 12], $x' = x + \epsilon$, is any combination of an original input $x$ and some small perturbation $\epsilon$, such that $f(x') \neq y_{\text{true}}$. This

definition is quite broad as it trivially includes inputs that $f$ simply fails to classify correctly. Generally, we are interested in the cases where $f(x) = y_{\text{true}}$, forcing $\epsilon > 0$, and will be addressing these in this work.

**Definition 2.6.** A **targeted** attack is an adversarial example such that $f(x') = y_{\text{target}}$ for some target label, $y_{\text{target}}$, that we choose. An **untargeted** attack is an adversarial example where we do not pick $y_{\text{target}}$ and simply achieve $f(x') \neq y_{\text{true}}$.

## 2.3 Threat Models

**Definition 2.7.** A **white box** attack is when the attacker has access to the parameters and architecture of the model. A **black box** attack is the opposite, where the attacker knows nothing about the internals of the model except its input and output specification. Unless explicitly stated, the attacks detailed are in a white box setting. This is due to the transferability property of adversarial examples which we will address in the black box attack (3.1.6). In general an attack needs only the ability to query the black box as a function at some reasonable rate, which we reasonably assume to be provided, to construct an adversarial attack on it.

## 2.4 A Brief on Optimization

### 2.4.1 Optimization Problem of a Targeted Attack

We first recall the initial proposed notion of a targeted attack on a classifier $f$, input $x \in D$, perturbed input $x' = x+\epsilon$, and target label $y_{\text{target}}$, as an optimization problem [12].

$$\begin{aligned} \underset{x'}{\text{minimize}} \quad & d(x, x') \\ \text{subject to} \quad & f(x') = y_{\text{target}} \\ & x' \in D \end{aligned}$$

A more specific formulation used in the typical case of soft labels is

$$\begin{aligned} \underset{x'}{\text{maximize}} \quad & P(y_{\text{target}}|x') \\ \text{subject to} \quad & d(x, x') \leq \epsilon \\ & x' \in D \end{aligned}$$

### 2.4.2 Lagrangian Relaxation

One common theme in the loss functions for generating adversarial examples is the utilization of the Lagrangian relaxation technique. Lagrangian relaxation is the process of relaxing a typical "hard" constraint and introducing it as a "soft" constraint in the objective function with the Lagrangian multiplier denoted as $\lambda$. It is an easier version of the problem, due to both an expanded search space and more "allowable paths". For example in the case of images, a hard constraint on the norm could restrict the optimizer to a local minimum despite there existing a path to the solution which goes through the unrestricted space. Moreover, the solution to the relaxed version of the problem approximates that of the original problem, and in practice, performing a line search (typically binary search of sorts) on $\lambda$ yields the optimal solution. Finally, we note that the terms that the Lagrangian multipliers are attached to do not matter much so long as the ratio between the terms remain the same. We will make this more concrete in the following examples.

# Chapter 3

# Attacks and Defenses

## 3.1 Selected Attacks

### 3.1.1 The Original (Unnamed) Targeted Attack

Szegedy *et al.* [12] originally addressed adversarial examples in the neural network space and proposed the following optimization problem and its Lagrangian relaxed version to generate them.

$$\begin{aligned} \underset{x'}{\text{minimize}} \quad & ||x - x'||_2 \\ \text{subject to} \quad & f(x') = y_{\text{target}} \\ & x' \in D \end{aligned}$$

Lagrangian Relaxed:

$$\begin{aligned} \underset{x'}{\text{minimize}} \quad & \lambda|x - x'|_2 + \text{loss}_f(x', y), \\ \text{subject to} \quad & x' \in D \end{aligned}$$

Here, the hard constraint of $x'$ being classified as $y_{\text{target}}$ directly correlates with the minimization of $\text{loss}_f(x', y)$, and for some threshold $\beta$

$$\text{loss}_f(x', y) \leq \beta \implies f(x') = y_{\text{target}}$$

Intuitively, $\lambda$ is simply a scaling factor for how severely we wish to penalize the magnitude of the perturbation and is used as a counterweight to the consideration of $\text{loss}_f$. If $\lambda = 0$, then we would expect $x' = x + \epsilon$ to be $f$'s interpretation of $y$ which, assuming $f \sim F$, would be rather useless as $x'$ is likely clearly distinguishable from $x$. On the other hand, if $\lambda$ is arbitrarily large then we severely limit our exploration space of $\epsilon = ||x - x'||$ (the adversarial perturbation layer) causing $f(x') = y$ to be unsatisfiable as the optimizer would be unwilling to explore a space which contains the solution. Of course, the value of $\lambda$ as a counterweight is dependent on the distribution of values that $\text{loss}_f$ can take.

### 3.1.2 Fast Gradient Sign Method (FGSM)

For the purposes of adversarial training (3.2.1), Goodfellow *et al.* [16] proposed a single step untargeted adversarial example generator. We note explicitly that this method is not intended to produce optimal adversarial examples (of minimal norm and maximal classification probability) but rather is intended to quickly produce adversarial examples for defensive training.

$$x' = x + \epsilon[\text{sign}(\nabla_x \text{loss}_f(x, y_{\text{true}}))]$$

Simply put, in our input image, $x$, we take a step of $\epsilon$ magnitude away from the direction of the gradient which classifies $x$ as $y$. Goodfellow *et al.* mention that the surprising effectiveness of a single step attack is due to the inherent underlying linearity of neural networks.

Adversarial attacks are typically split into single step attacks and iterative attacks, with the former being more efficient and transferable [17] for a trade-off of optimality in the attack to the latter. One possible explanation for this is that iterative methods, being more explorative than single step methods, converge on local minima unique to certain models whereas the single step method converges on a more general, local minima cluster which is specific to the class at hand.

### 3.1.3 iFGSM (Basic Iterative Method), iFGM, Projected Gradient Descent (PGD)

We discuss a number of variants of FGSM which have been popularized and their respective uses. Kurakin *et al.* [17] proposed the Basic Iterative Method (popularly known as iFGSM), an iterative version of FGSM which simply reduces the step size to some new value $\alpha$ and clips the resulting tensor by $\epsilon$ distance from the original image, assigning $x$ in the next step to be the $x'$ computed in the previous step. So, the total magnitude of difference is still limited to $\epsilon$ but iterative sampling of the gradient allows for more optimal results.

$$x'_0 = x, \quad x'_{n+1} = \text{clip}_{x,\epsilon}(x'_n + \alpha \text{sign}(\nabla_{x'_n} \text{loss}_f(x'_n, y_{\text{true}})))$$

We briefly mention FGM and iFGM which are simply the unsigned versions of FGSM and iFGSM, i.e. they step along the gradient. Moreover, the targeted versions of these attacks are created by substituting $y_{\text{true}}$ for $y_{\text{target}}$ and negating the gradient (as we are now stepping to the target label rather than stepping away from the true label).

In essence, these iterative methods are just instances of projected gradient descent, where the clipping of the output by $\epsilon$ enforces a constraint of $d(x, x') \leq \epsilon$. This is notable as Madry *et al.* [18] presents empirical evidence for why PGD should be considered a universal first order adversary, that is, no other adversaries which merely depend on the gradient should expect to do significantly better than PGD.

Thus, Madry *et al.* claim that training a network to be robust against PGD attacks should implicate its robustness against a wide-range of first-order attacks.

### 3.1.4 Carlini & Wagner

Carlini *et al.* [19] propose an attack which is similar to the one proposed by Szegedy *et al.* but replaces the loss function with a transformation of the classifier (or more specifically a function of either its softmax layer or its logits layer) parametrized by a label $y$, call it $f_y^T(x)$, with the property that $f_y^T(x) \leq 0$ if and only if $f(x) = y$. Carlini *et al.* propose numerous choices for $f^T$, selecting the best one from empirical testing. As the selection of $f^T$ as an adversarial example optimization specific loss function is one of the main features of this attack, we refer the reader to their paper for further details on its selection.

$$\begin{aligned} \underset{x'}{\text{minimize}} \quad & d(x', x) + \lambda f_y^T(x'), \\ \text{subject to} \quad & x' \in D \end{aligned}$$

### 3.1.5 Interesting Constrained Attacks: One Pixel Attack and Universal Adversarial Perturbations

We briefly review, on a high level, some more interesting attacks with specific properties and encourage the reader to read the respective papers for more detail.

One might wonder what a lower bound for the number of modified pixels required for mounting a successful attack may be. By constraining the $l_0$ distance, i.e. the number of pixels modified, to be no more than one in the optimization procedure, Su *et al.* [20] show not only that it is possible with one pixel but that it is probable as well.

$$\begin{aligned} \underset{x'}{\text{maximize}} \quad & P(y_{\text{target}}|x') \\ \text{subject to} \quad & ||x, x'||_0 \leq 1 \\ & x' \in D \end{aligned}$$

While the feasibility of the attack is fairly reduced in doing so, both in the number of possible target classes for a given input and the confidence that can be invoked in the classifier, it is still surprising that over 65% of CIFAR-10 and over 40% of ImageNet validation images were susceptible to this attack under common network architectures for those tasks [20].

Another question one might wonder is if there exist general adversarial perturbations which cause misclassifications regardless of the input they are superimposed on. With an adversarial perturbation layer $v$ generated on a distribution $X$, define the fooling rate $M(X, v)$ as an extension of the misclassification rate for a specific perturbation layer as:

$$M(X, v) = \sum_{x \in X} \frac{\mathbb{1}_{\text{true}} f(x + v) \neq y_{\text{true}}}{|X|}$$

That is, the percentage the misclassifications that $v$ incurs on $X$. For a $\delta$ we pick, say that $v$ is universal on $X$ if $M(X, v) \geq 1 - \delta$.

Moosavi-Dezfooli *et al.* [21] formulated the following algorithm to generate universal adversarial perturbations with the constraint that $||v||_k < \epsilon$.

1. initialize $v = 0$
2. while $M(X, v) \leq 1 - \delta$
   (a) for each $x_i \in X$
      i. compute the minimal $v_i$ by a chosen norm metric s.t. $f(x_i + v + v_i) \neq y_{\text{true}}$
      ii. set $v = clip_\epsilon(v + v_i)$

In short, Moosavi-Dezfooli *et al.* [21] showed that a greedy averaging of adversarial perturbations constrained in a projection space of radius $\epsilon$ can be individually adversarial for each of the samples it averaged on. Notably, for ImageNet classifiers and $|X| = 10000$ (on average 10 pictures per class), they report at least an 80% fooling rate on both the initial set $X$ and the validation set (50000 images). Perhaps more surprising is the attaining of an above 30% fooling rate on the validation set with $|X| = 500$, fooling the classifier even on images of classes that were not included in $X$. Figure 3.1 shows an example of a universal adversarial perturbation from their results.

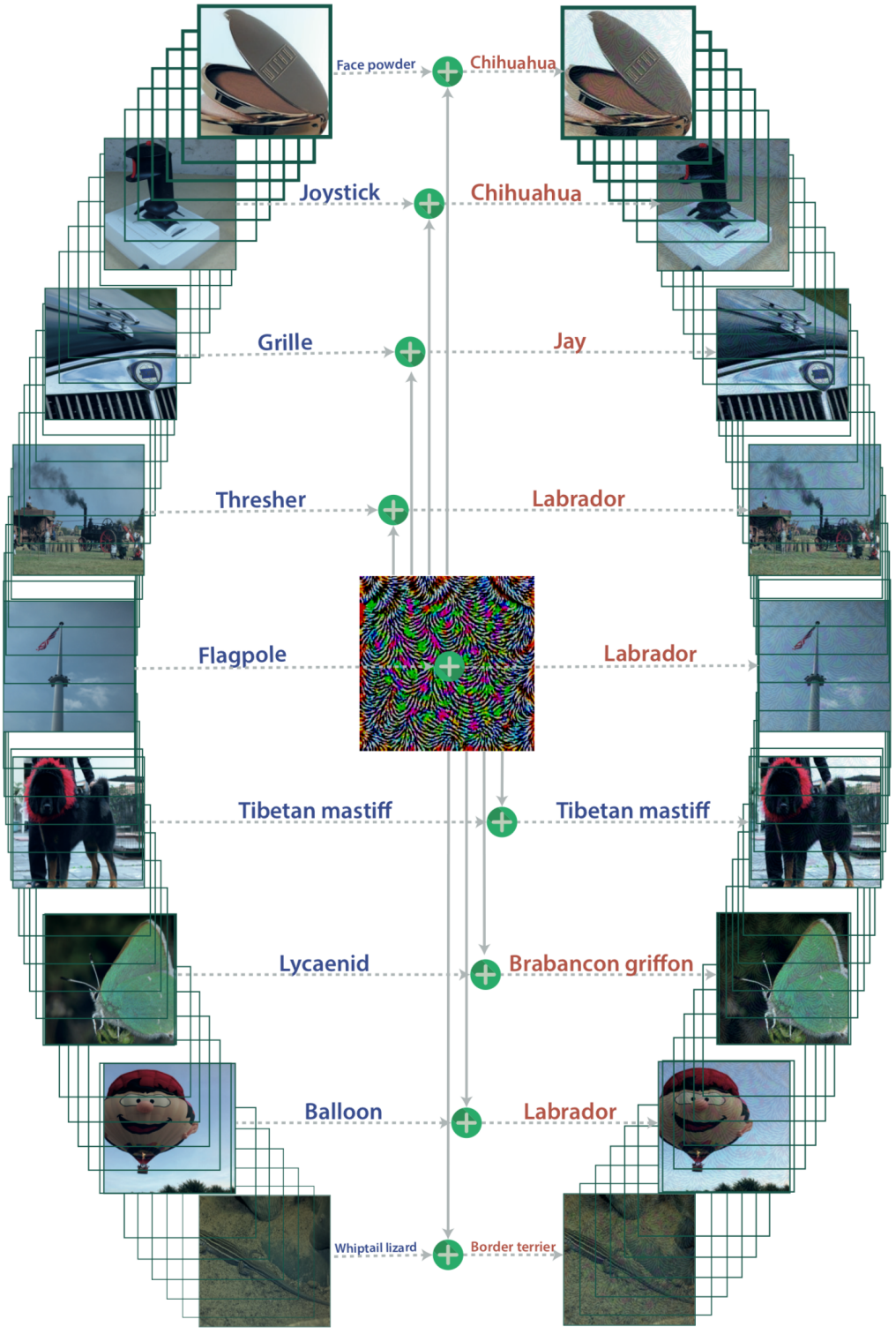



FIGURE 3.1: Universal Adversarial Perturbation Example
*Left*: Original Image
*Middle*: Universal Adversarial Perturbation
*Right*: Adversarial Image
Moosavi-Dezfooli *et al.* [21]

### 3.1.6 Black Box Attacks

Due to the transferability property of neural networks, the attacks we have described and most of the attacks introduced are in the context of a white box threat model. The concept of transferability is that different classifiers which encompass the same input-output pairs ultimately approximate the same function. Of course, the functions are the same if the input-output pairs cover the whole domain and codomain-spaces but empirical results show that the approximation is still noticeable even under small subsampling. For example, classifiers which perform the same task are thought to learn approximately equivalent features. This leads to adversarial examples generated on a particular classifier trained on a particular training set still being adversarial for different classifiers of the same problem trained on different training sets.

Thus, given a black box classifier $f$, an attack would first architect and train their own white box model on the input-label pairs $(x, f(x))$, generate adversarial examples using some chosen white box attack, and apply that example to $f$. The architecture of $f$ can typically be posited from the problem space $f$ is in (say convolutional networks for spatial problems, recurrent networks for temporal problems, etc.) and the training data can be generated similarly, the only requirement being that the attacker can reasonably sample $f$ to train their model [11, 12]. In fact, excluding the One Pixel attack, the attacks above report successful transferring of adversarial examples.

### 3.1.7 Backward Pass Differential Approximation (BPDA)

Certain defenses involve inserting a preprocessor layer, $g$, to "clean" the input of adversarial impurities before passing it along to the classifier, $f$. If $g$ is smooth and differentiable, then any attack on $f$ can be modified to work on $f \circ g$ by inserting $g$ in its forward pass and $\nabla_x g(x)$ in its backpropagation pass. However, if $g$ is neither smooth nor differentiable and $g(x) \approx x$, Athalye *et al.* [22] propose

to approximate $\nabla_x g(x) = \nabla_x x = 1$ in the backpropagation step, and the attack is shown to still succeed.

### 3.1.8 Expectation over Transformation (EOT)

Thus far, the adversarial examples we have discussed have remained in the digital realm. In the following defenses section, we see that adversarial examples are spatially fragile, that is, the generated adversarial perturbation layer relies on a fixed spatial component of the underlying image. Then, attempting to transfer these adversarial examples to the physical world incurs various forms of spatial transformations which mitigate the adversarial nature altogether. Athalye *et al.* [14] proposed the idea of modelling the distribution of transformations on a digital image to a physical image. For this purpose, the EOT algorithm generates adversarial examples which survive such a distribution of transformations. In particular, given a distribution of transformations $T$, we approximate the expectation of the loss function of our adversarial example by sampling it over transformations $t \leftarrow T$.

$$\arg\min_{x'} E_{t \leftarrow T}[\text{loss}_f(y_{\text{target}}, t(x')) + \lambda d(x, x')]$$

Extending the algorithm, Athalye *et al.* [14] were able to approximate the transformation from the digital to real world as a composition of differentiable transformations and consequently showed that they could generate 3-D adversarial objects, print them, and have them maintain their adversarial nature from viewing conditions within the distribution of transformations they modeled.

## 3.2 Selected Defenses

As the attacks themselves are fairly recent, the defenses are also in development, with many of them having been broken already [22–24]. To our knowledge, there

does not yet exist a robust defense against adversarial examples. Although the defenses mentioned below have been broken, there is a persisting theme of attempting to "hide" the problem. That is, instead of generally fixing the adversarial classification spaces to be classified as the correct label, the current defenses attempt to hide them, either by means of gradient masking (rendering the gradient unusable in some sense) or relocating the adversarial spaces by some selection of transformations.

### 3.2.1 Adversarial Training (Hardening)

Recall that Goodfellow *et al.* [16] proposed the FGSM attack to efficiently generate adversarial examples for the purpose of adversarial training (sometimes called adversarial hardening). Adversarial training is simply the incorporation of adversarial examples into the training set of the classifier. However, as Kurakin *et al.* [25] found, such training only patches individual adversarial regions resulting in a local smoothing of the decision boundary and gradient. While the gradients local to valid classification spaces may be unusable, the positions of the adversarial examples outside the locally "patched" space remain unchanged. As Kurakin *et al.*, Papernot *et al.*, and Athalye *et al.* have discovered [22, 25, 26], gradient masking merely hides the problem and does not fix it.

General gradient masking methods are prone to black box transfer attacks by generating an attack outside of the locally smooth gradient on a substitute model. Figure 3.2 shows an example in one dimension.

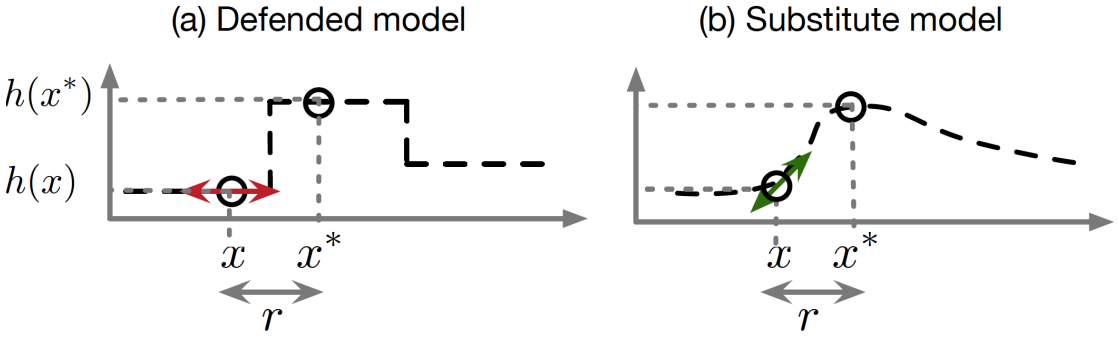



FIGURE 3.2: Papernot *et al.* [27] illustrate the 1-D case

As an aside, Tramèr *et al.* [25] bypass local gradient masking by modifying the FGSM algorithm to initialize with a step in a random direction of some small magnitude sufficient enough to escape the local smoothing.

### 3.2.2 Defensive Distillation

Just as humans learn better through the seasoned intuition of effective teachers, Hinton *et al.* proposed distilling the learned information from some trained heavy (computation wise) deep model teacher $M$ to some lighter more shallow model learner $M'$ by training $M'$ on the output of $M$'s softmax layer [28]. The underlying intuition is that the perceived probability distribution of labels for each training input is vastly richer than the hard ground truth label that is originally provided to the teacher, and in fact contains the learned details necessary for the classification task.

Recall the softmax function for an input vector, $\langle x \rangle$. We introduce a new parameter $T$ for the temperature of the softmax function.

$$\sigma_x(x_\ell) = \frac{e^{x_\ell}/T}{\sum\limits_{x_\ell \in x} e^{x_\ell}/T}$$

$T$ can be described as a measure for how "soft" the output probabilities are. Specifically, a higher $T$ will "smooth" the labels, i.e. cause a lower variance in the probabilities as their exponential differences are scaled down and thereby prevents the learner from overfitting [28]. Of course, the value of $T$ is relative to the magnitude of values in $\langle x \rangle$. For the distillation method, $M'$ is trained with $T > 1$ but restored to $T = 1$ for validation and classification such that it produces "harder" labels.

The defensive distillation method proposed by Papernot *et al.* [29] is the distillation method described above with an abnormally high temperature during training. Put simply, smoothing of the labels in the learner $M'$ has the secondary effect of smoothing the gradient in the space of each label. Then, an abnormally

high temperature smooths the gradient so much so that any gradient based adversary is presented with a useless gradient on any valid classification in the white box attack model.

While Carlini *et al.* [30] successfully attacked the white box model by artificially reinstating the training temperature into the classifier, thereby restoring the gradient, we note that because this is a gradient masking defense it is generally prone to black box attacks [26, 27].

### 3.2.3 Random Transformations

As adversarial attacks typically target specific minima which are close to the examples, perhaps transforming the input space will be able to move or altogether remove the minima. Guo *et al.* [31] propose transformations such as bit-depth reduction (rounding pixel values to omit the least significant bits), JPEG compression and decompression, variance minimization, etc. for this purpose. In exploiting the spatial dependency of adversarial perturbations, such defenses seemed to work. Xie *et al.* [13] similarly concluded that random resizing and random padding successfully mitigate adversarial attacks.

Recall from the attacks section that the EOT (3.1.8) algorithm bypasses transformations such as resizing, random padding, scaling, etc. Then, JPEG compression, bit-depth reduction, and variance minimization are all transformations which approximately minimally alter the input image, thus the BPDA (3.1.7) algorithm bypasses these defenses.

### 3.2.4 Feature Squeezing

The defenses mentioned thus far have relied on modifying the neural network itself in some sense. Xu *et al.* [32] propose a defense with external detectors which operate on a lower fidelity or "squeezed" version of the input. They proposed bit-depth reduction and spatial smoothing (convolutional blurring) to reduce the size

of the input space thereby reducing the feasibility of adversarial examples. The resulting classifier outputs a label only if the differences between the original classifier's outputted probabilities of that label between the input and its "squeezed" versions are below a certain threshold.

Under the stronger threat model of an *adaptive adversary*, one which knows about the defenses (in this case, the squeezing transformations) present in the network, He *et al.* [33] propose bypassing each detector individually and composing those attacks. In both detectors, they employ the Carlini & Wagner attack (3.1.4). For the bit-depth reduction, they simply randomize multiple starting points for the attack. For the spatial smoothing, they employ a median filter (smoothing filter) inside the classifier itself such that their adversarial example is generated under a model that smooths the image. For the composition of these attacks, we refer the reader to the original work.

### 3.2.5 Ensemble Adversarial Training

We stated earlier that due to the transferability of black-box attacks, the threat model most commonly considered is the white-box threat model. Tramèr *et al.* [25] propose a defense purely for black-box attacks by training the undefended model on adversarial perturbations generated from other, equivalent models. With the initial goal of "decoupling" adversarial example generation from the undefended model, Tramèr *et al.* simultaneously provide a solution for scaling adversarial training by establishing the efficacy of predefined adversarial training sets for a given problem domain as well as increasing robustness to black box attacks in exposing the undefended model to weaknesses of other models.

One surprising result is that ensemble adversarial training did not help at all with defending against white box constructed attacks. One interpretation of this is that the minimum distance to the space of transferable adversarial examples is noticeably greater than that of white box adversarial examples, which is unsurprising as transferable adversarial examples satisfy strictly more constraints. Then, perhaps

it would make sense to distinguish not only attacks, but also the spaces of inputs that they exploit, by the threat models they originate from.

# Chapter 4

# Experiments and Findings

## 4.1 Motivation

Among the many technologies that adversarial examples pose a serious threat to, autonomous vehicles is thought to be one of the most critical. However, this concern was momentarily relieved when Lu *et al.* [34] demonstrated that adversarial stop signs printed into the physical world posed no real threat due to the real world noise incurred on them. Particularly, various factors such as distance, lighting, viewing angle, etc. rendered the adversarial nature undetectable by the sensors on the autonomous car, leading Lu *et al.* to conclude that adversarial examples, as they were, did not pose a serious threat to autonomous vehicles.

Additionally, Kurakin *et al.* [15] showed the persistence of 2-D adversarial examples under what they coin as a black box "photo transformation", i.e. the process of printing adversarial examples and taking a photo of it, in a somewhat controlled environment. Specifically, they would take pictures of the printed photos without careful control of lighting, camera angle, camera distance, etc., but did employ perspective transform to the captured image restoring it to its original dimension and orientation before the photo transformation. Moreover, they examined and plotted the destruction rate of adversarial examples with respect to various digital transformations, such as brightness, contrast, noise, blur, and JPEG encoding.

With EOT (3.1.8), we were able to bypass all of the aforementioned destructors of adversarial examples. Such results are in line with Athalye *et al.* [14], with the exception of our bypassing of JPEG encoding and decoding, which Athalye *et al.* did not comment on.

### 4.1.1 Distribution Selection

We ran our simulations with the following composition of random transformations used by Athalye *et al.* [14]:

| transformation | min | max |
|---|---|---|
| scale | 0.9 | 1.4 |
| rotate | $-22.5°$ | $22.5°$ |
| lighten/darken | -0.05 | 0.05 |
| Gaussian noise (stdev) | 0 | 0.1 |
| translation | -40px | 40px |

We also experimented with subsets of this selection on a wider range of transformation values and were successful. However, as the output space of the distribution grows roughly exponentially with the number of dissimilar transformations composed, it is necessary to reduce the bound of magnitude of each transformation for the problem to remain tractable.

### 4.1.2 Methodology

The training runs consisted of a Tensorflow implementation run on a Linux distribution using a single GTX1070. For the photo transformations, we used a Samsung Galaxy S7 12MP camera. For printing, we used a standard color printer. For monitor displays, we used an IPS (known for their color accuracy) monitor and a MacBook Pro Retina display.

We refer to 4.1.3 for the details of parameter selection and thus omit such details in the methodology.

Our steps were as follows:

1. Pick some arbitrary starting class from ImageNet and manually find a corresponding image with Google image search
2. Pick some arbitrary target class from ImageNet, $y_{\text{target}}$
3. Generate an targeted adversarial example, $x'$, on the chosen target class using EOT
   a. We chose the well-known InceptionV3 architecture by Sgezedy *et al.*[35] as our classifier $f$, importing the model[1] and pretrained weights[2].
   b. For image preprocessing, we down-scaled the image such that either the height or the width was 299px. Then, we cropped to 299px $\times$ 299px, the input shape for InceptionV3, from the top left corner.
   c. As no implementation was available, we implemented EOT as detailed by Athalye *et al.* [14], with stochastic gradient descent as prescribed in their work.
4. We verify the adversarial example persists under the composition of random transformations
   a. We successively apply the individual transformations in-order, from our selected distribution (with the magnitude sampled uniformly from the ranges listed), on $x'$ to generate $x'_T$. We then check if $y_{\text{target}}$ is in the top-1 in the output of $f(x'_T)$, verifying that EOT succeeds as expected [14].
5. We verify the adversarial example persists under photo transformation from printout and from screen, and further transformations afterward. The images are printed and displayed at approximately 150 pixels-per-inch as we found that

[1] `https://github.com/tensorflow/tensorflow/blob/master/tensorflow/contrib/slim/python/slim/nets/inception_v3.py`

[2] `http://download.tensorflow.org/models/inception_v3_2016_08_28.tar.gz`

any less results in destruction of the adversarial example. We detail the photo transformation process and note that the verification methods are the same as the previous step.

a. Take a picture of the image straight on, in a reasonably lit environment, and without flash.

b. On the phone, manually crop to a square aspect ratio, maintaining the image inside.

c. Image is automatically compressed as a JPEG file.

d. Send the file to self over Facebook messenger, which results in further JPEG compression.

e. Save file from messenger to computer.

f. Run the verification procedure of 4.a. with our photo transformed image as $x'$.

As some factors, such as contrast, viewing angle, color inaccuracy, unsaturated colors, glare, etc., outside of our chosen distribution due to a lack of incorporation. Thus, we attempted taking pictures directly from the screen to try and align with our distribution more. Particularly pictures taken from the screen were from different monitors, one glossy and full of smudges and another matte and clear, under different brightness levels, and under different lighting levels, with some having my shadow reflected on them. Additionally, the pixelation, light bleeding, and vertical synchronization problems appeared on digital shots as well. Overall, the adversarial images were fairly resilient under this transformation and managed to maintain top-1 targeted classification. We show a selection of the best results in appendix A.

Due to a constraints of both computational power and time (generating one adversarial example under the setting described required at least several hours), and the manual efforts required, we were unable to produce many results.

### 4.1.3 Parameter Tuning

1. For an $L_2$ norm in the loss function, we found $\epsilon = 1e^{-2}$ to be the ideal learning rate in achieving both convergence and efficiency. For a more optimal solution in terms of reduced norm distance, we found that $\epsilon = 1e^{-3}$, directly recommended by Athalye, still converges, performing better but at the cost of an order of magnitude more of training time.

2. For sample size, we found that at least $n = 20$ samples were needed to have "useful" steps towards convergence, and $n = 50$ samples to be a good balance of meaningful steps and efficiency.

3. While the Lagrangian constraint is dependent on the input itself, we found the range of optimal values to be around $0.01 \leq \lambda \leq 0.03$, choosing $\lambda = 0.015$ for initial training and manually fine tuning when convergence halts. The manual tuning consisted of raising $\lambda$ if the classification probability of the target was above 90% but the norm distance was more than desirable, and lowering $\lambda$ if the classification probability was below 90%.

### 4.1.4 Informal Findings

1. With regards to standalone transforms, the brightness transform was the easiest to bypass and Gaussian noise, the hardest, by measure of required sample size and number of training steps.

2. Just as word embeddings have given a spatial representation to the distance between words, minimal distances between distributions of images by label roughly correlate. For example, it's much easier to find targeted adversarial examples from one furry animal to another furry animal than it is from one furry animal to some metal object. Intuitively, this roughly scales with the choice of the Lagrangian constraint, i.e. the farther away the adversarial example is expected to be, the smaller the constraint on the norm should be.

3. JPEG encoding and decoding was a destructor for adversarial examples of single transformations, but adversarial examples under the aforementioned composition were resistant to JPEG encoding and decoding. Specifically, it seemed like hardening adversarial examples against Gaussian noise transformation led to their hardening against JPEG encoding and decoding.

4. As Carlini *et al.* For adversarial examples within a small $\epsilon$ ball, integrality becomes a substantial factor as the values are trained in a space of $[0, 1] \in \mathbb{R}$ but saved to $[0, 255] \in \mathbb{Z}$. This can be overcome either by increasing the perturbation distance or by specifically training against integrality transformations. The former is simpler to do and in practice, the minimum satifisable $\epsilon$ for EOT under a sufficiently complex distribution is large enough s.t. integrality does not need to be considered. However, this was somewhat of an issue under single transformations where the norm loss of the adversarial example was less than a integral unit.

5. Typically, the square norm is preferred due to two reasons. One, the square norm penalizes quadratically more as the norm increases. This corresponds to perception of change in pixel values being increasingly more important after a certain threshold. Additionally, there are numerical instability issues with the norm when it is equal to 0 in certain libraries, particularly tensorflow. Thus explaining why the squared norm has been widely popularized.

6. Our average $L_2$ loss per pixel was no more than $4e^{-2}$. The discrepancy from Athalye *et al.*'s average loss of $5.6e^{-5}$ can be attributed to our learning rate being an order of magnitude greater.

### 4.1.5 An Extension of EOT

Athalye *et al.* show that EOT generated adversarial examples successfully persist in transformations selected from the chosen distribution they were trained on but explicitly makes no guarantees for cases that transformations are not in the

distribution. Empirically, we found the persistence of EOT generated adversarial examples under some transformation to be biconditional with that transformation being in the training distribution, that is, the transformation being from the distribution for persistence of the adversarial nature is not only sufficient, it is necessary.

Thus, we propose the concept of an "activated" adversarial example. In particular, where previous applications of EOT have always included the identity transformation inside the chosen distribution, we can intentionally choose to omit the identity, and the neighboring space of transformations around it, from the training distribution to generate an adversarial example which is adversarial only under a distribution of our choice.

As a hypothetical example, if we decided to target roadway signs, we could use EOT to generate our adversarial stop sign under a certain lighting condition which is atypical. Then, a self driving car would recognize it as a stop sign most of the time, except when our chosen atypical lighting condition is present, the car would fail to recognize the stop sign.

In this manner, an attacker would be able to dictate the conditions under which their attack was adversarial, rendering an arguably more troublesome adversary.

As our testing was on simple transformations and verified digitally, we encourage further work on extending this to compositions of transformations verified in the real world.

### 4.1.6 Further Work

We recommend the Cleverhans library [36] to any readers who may be interested in trying their hand in this. Additionally we have released our code[3] for those interested.

[3] `https://github.com/steven200796/Synthesizing_Adversarial_Examples`

# Appendix A

## A.1 Cat to Guacamole

### A.1.1 Initial Test

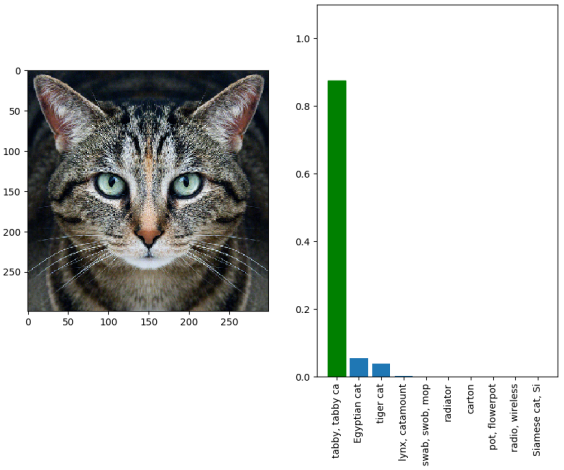



FIGURE A.1: Initial Image

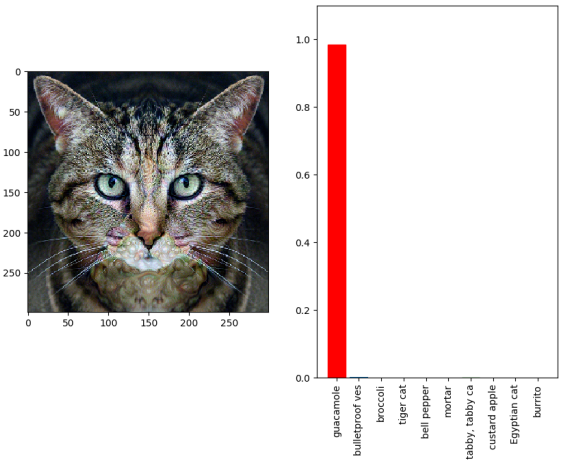



FIGURE A.2: Adversarial Example

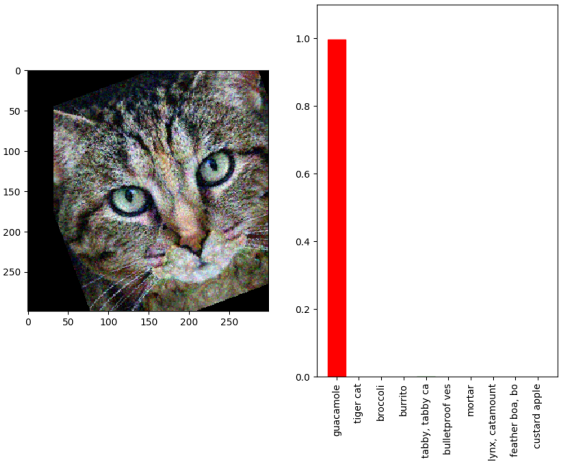



FIGURE A.3: Composition of Transformations 1

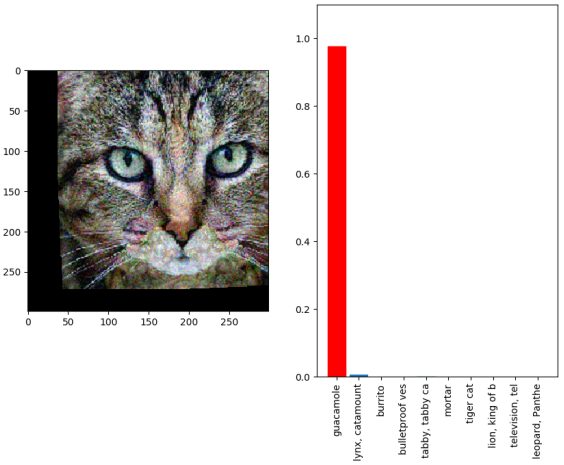



FIGURE A.4: Composition of Transformations 2

### A.1.2 Photo Transformation from Printout

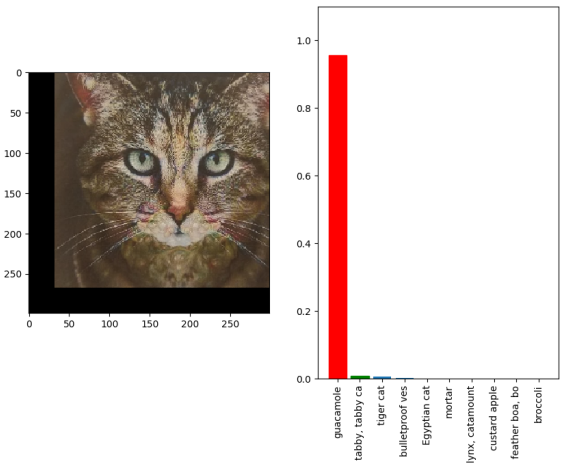



FIGURE A.5: Printout Translated

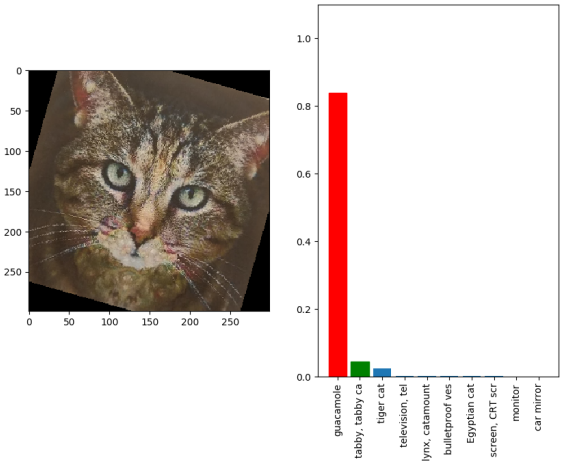



FIGURE A.6: Printout Rotated

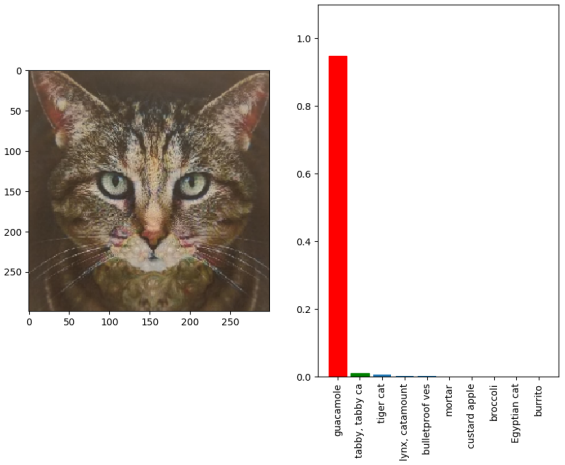



FIGURE A.7: Printout Scaled

### A.1.3 Photo Transformation from Monitor

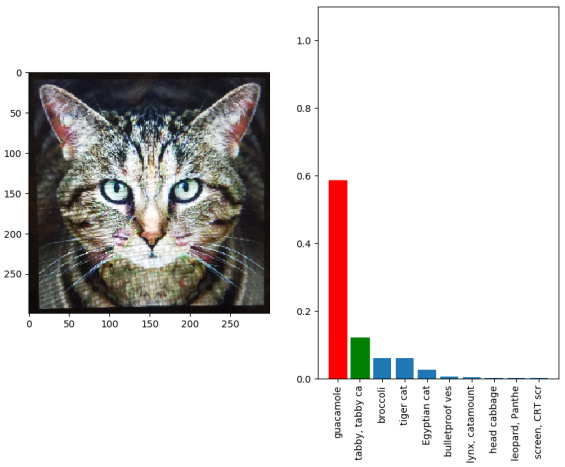



FIGURE A.8: Screen Brightened

## A.2 Chihuahua to Cockroach

### A.2.1 Initial Test

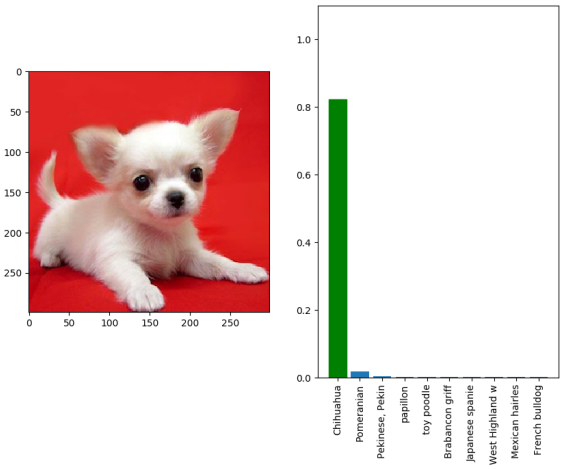



FIGURE A.9: Initial Image

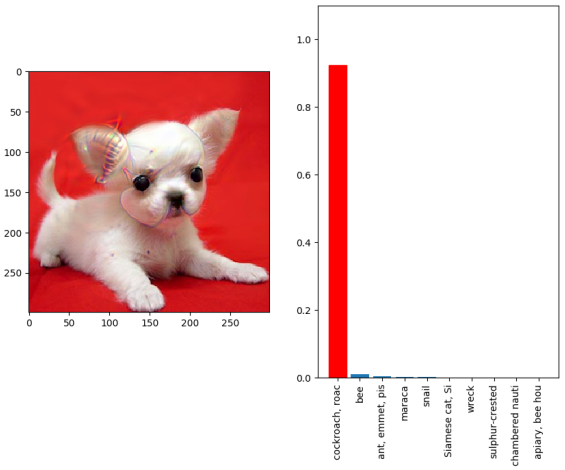



FIGURE A.10: Adversarial Example

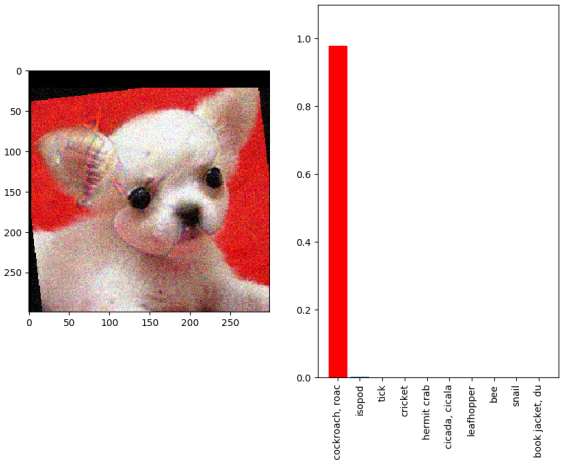



FIGURE A.11: Composition of Transformations 1

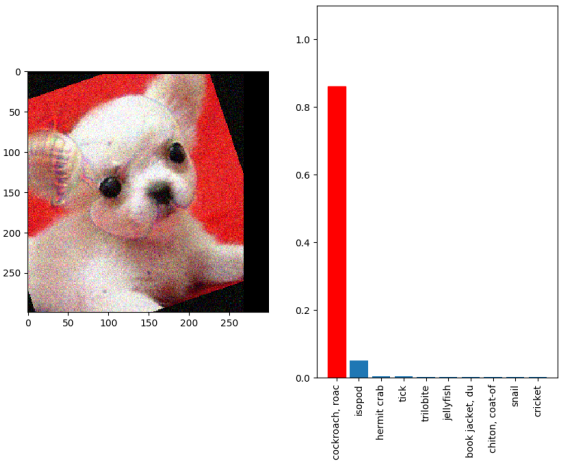



FIGURE A.12: Composition of Transformations 2

### A.2.2 Photo Transformation from Printout

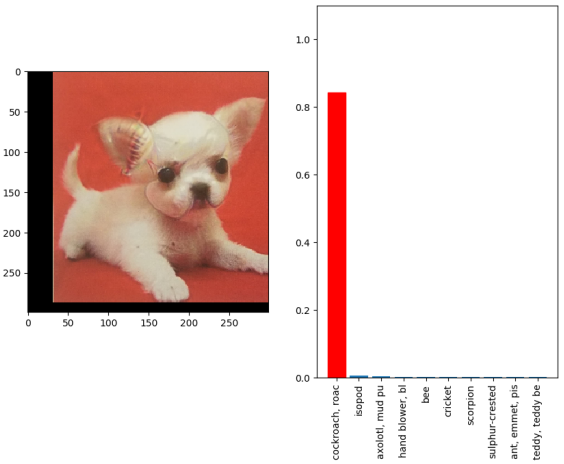



FIGURE A.13: Printout Translated

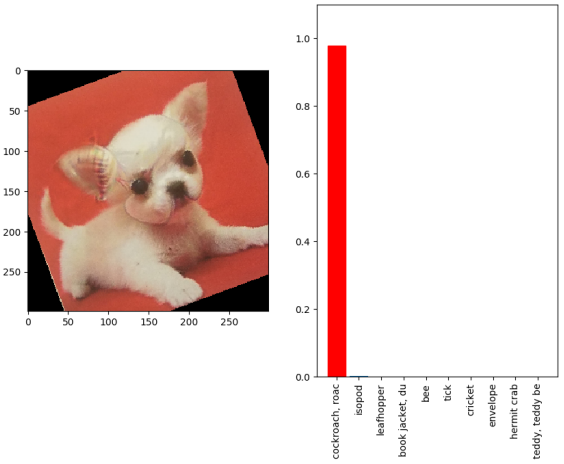



FIGURE A.14: Printout Rotated

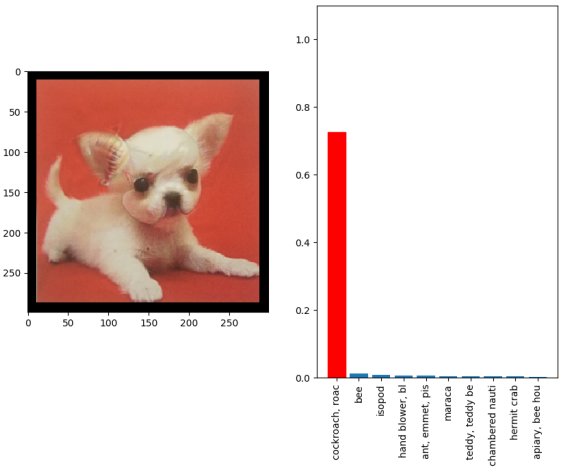



FIGURE A.15: Printout Scaled

### A.2.3 Photo Transformation from Monitor

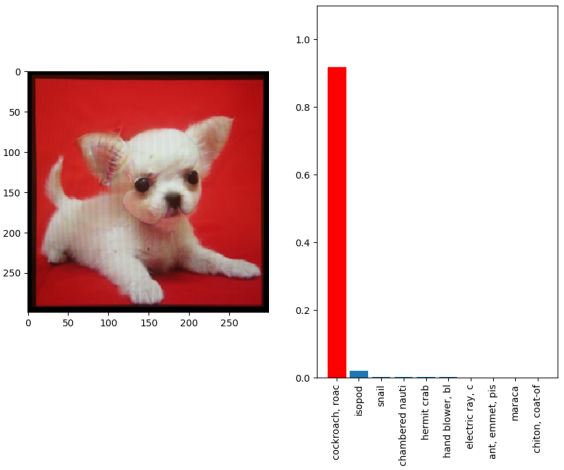



FIGURE A.16: Screen Scaled

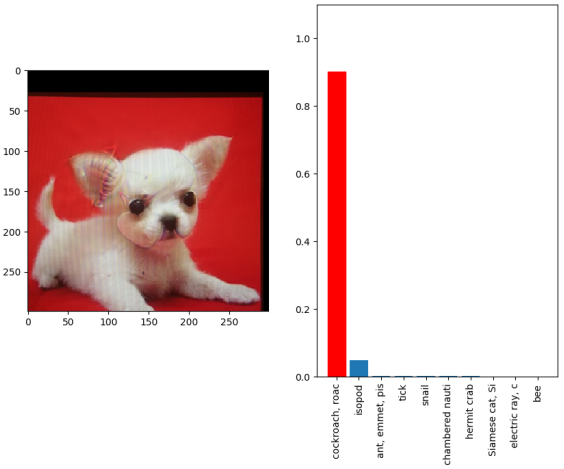



FIGURE A.17: Screen Translated

## A.3 Hot Pot to Pillow

### A.3.1 Initial Test

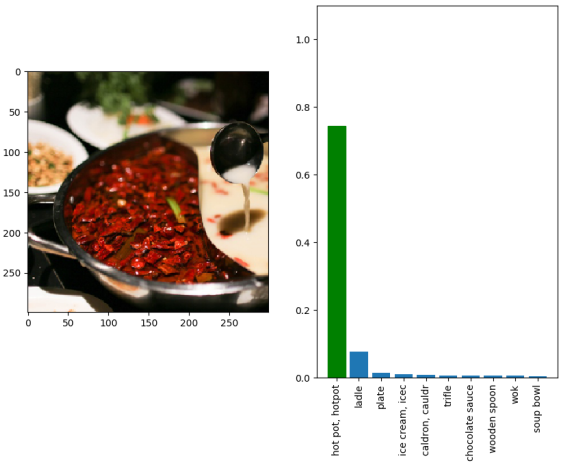



FIGURE A.18: Initial Image

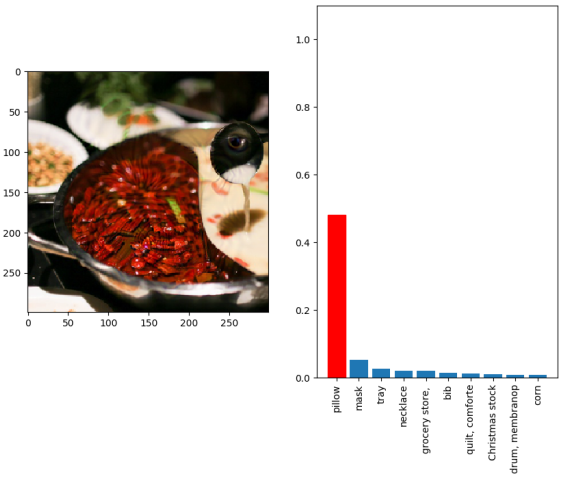



FIGURE A.19: Adversarial Example

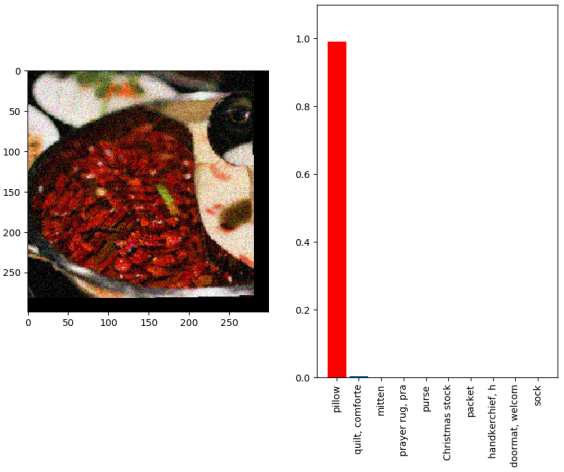



FIGURE A.20: Composition of Transformations 1

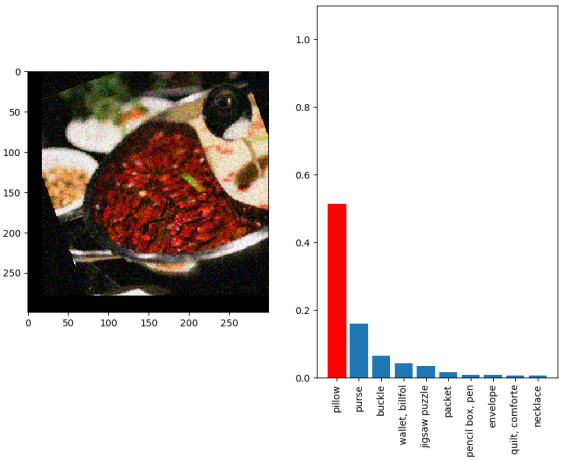



FIGURE A.21: Composition of Transformations 2

### A.3.2 Photo Transformation from Printout

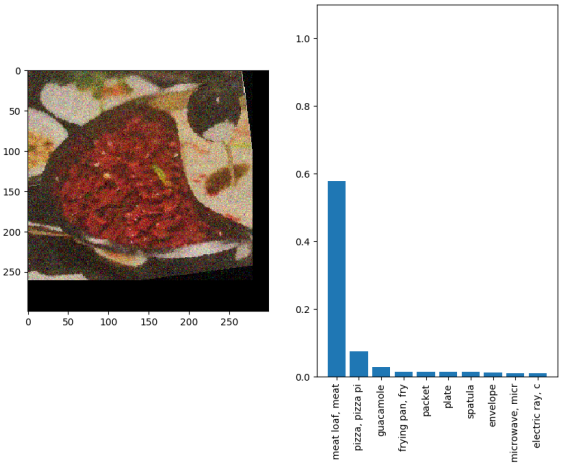



FIGURE A.22: Printout Composition of Transformations 1

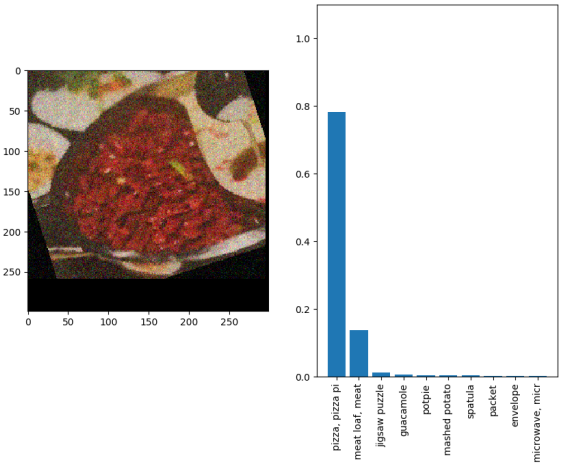



FIGURE A.23: Printout Composition of Transformations 2

### A.3.3 Photo Transformation from Monitor

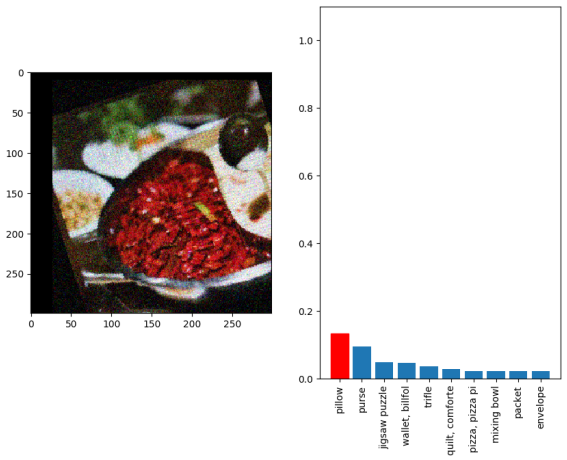



FIGURE A.24: Screen Composition of Transformations 1

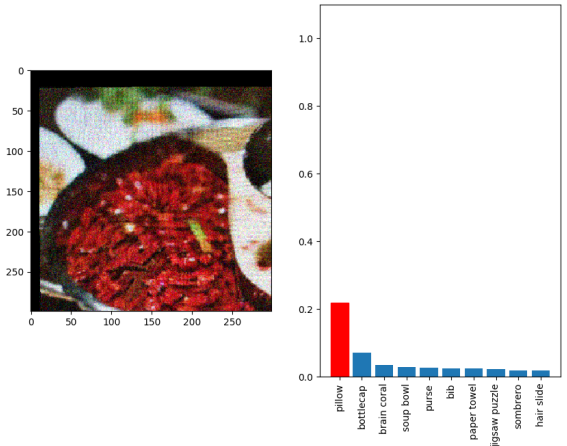



FIGURE A.25: Screen Composition of Transformations 2

# Bibliography

[1] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012. URL `http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf`.

[2] Justin Johnson, Andrej Karpathy, and Fei-Fei Li. Densecap: Fully convolutional localization networks for dense captioning. *CoRR*, abs/1511.07571, 2015. URL `http://arxiv.org/abs/1511.07571`.

[3] Geoffrey Hinton, Li Deng, Dong Yu, George Dahl, Abdel rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara Sainath, and Brian Kingsbury. Deep neural networks for acoustic modeling in speech recognition. *Signal Processing Magazine*, 2012.

[4] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Vedavyas Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy P. Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016. doi: 10.1038/nature16961. URL `https://doi.org/10.1038/nature16961`.

[5] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin A. Riedmiller, Andreas Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015. doi: 10.1038/nature14236. URL `https://doi.org/10.1038/nature14236`.

[6] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. A neural probabilistic language model. *J. Mach. Learn. Res.*, 3:1137–1155, March 2003. ISSN 1532-4435. URL `http://dl.acm.org/citation.cfm?id=944919.944966`.

[7] Oriol Vinyals and Quoc V. Le. A neural conversational model. *CoRR*, abs/1506.05869, 2015. URL `http://arxiv.org/abs/1506.05869`.

[8] Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew W. Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *CoRR*, abs/1609.03499, 2016. URL `http://arxiv.org/abs/1609.03499`.

[9] Scott E. Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. Generative adversarial text to image synthesis. *CoRR*, abs/1605.05396, 2016. URL `http://arxiv.org/abs/1605.05396`.

[10] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. A neural algorithm of artistic style. *CoRR*, abs/1508.06576, 2015. URL `http://arxiv.org/abs/1508.06576`.

[11] Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Srndic, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. Evasion attacks against machine learning at test time. *CoRR*, abs/1708.06131, 2017. URL `http://arxiv.org/abs/1708.06131`.

[12] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *CoRR*, abs/1312.6199, 2013. URL `http://arxiv.org/abs/1312.6199`.

[13] Cihang Xie, Jianyu Wang, Zhishuai Zhang, Zhou Ren, and Alan L. Yuille. Mitigating adversarial effects through randomization. *CoRR*, abs/1711.01991, 2017. URL `http://arxiv.org/abs/1711.01991`.

[14] Anish Athalye, Logan Engstrom, Andrew Ilyas, and Kevin Kwok. Synthesizing robust adversarial examples. *CoRR*, abs/1707.07397, 2017. URL `http://arxiv.org/abs/1707.07397`.

[15] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *CoRR*, abs/1607.02533, 2016. URL `http://arxiv.org/abs/1607.02533`.

[16] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*, 2015. URL `http://arxiv.org/abs/1412.6572`.

[17] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. Adversarial machine learning at scale. *CoRR*, abs/1611.01236, 2016. URL `http://arxiv.org/abs/1611.01236`.

[18] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *CoRR*, abs/1706.06083, 2017. URL `http://arxiv.org/abs/1706.06083`.

[19] Nicholas Carlini and David A. Wagner. Towards evaluating the robustness of neural networks. *CoRR*, abs/1608.04644, 2016. URL `http://arxiv.org/abs/1608.04644`.

[20] Jiawei Su, Danilo Vasconcellos Vargas, and Kouichi Sakurai. One pixel attack for fooling deep neural networks. *CoRR*, abs/1710.08864, 2017. URL `http://arxiv.org/abs/1710.08864`.

[21] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Universal adversarial perturbations. *CoRR*, abs/1610.08401, 2016. URL `http://arxiv.org/abs/1610.08401`.

[22] Anish Athalye, Nicholas Carlini, and David A. Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. *CoRR*, abs/1802.00420, 2018. URL `http://arxiv.org/abs/1802.00420`.

[23] Nicholas Carlini and David A. Wagner. Adversarial examples are not easily detected: Bypassing ten detection methods. *CoRR*, abs/1705.07263, 2017. URL `http://arxiv.org/abs/1705.07263`.

[24] A. Athalye and N. Carlini. On the Robustness of the CVPR 2018 White-Box Adversarial Example Defenses. *ArXiv e-prints*, April 2018.

[25] Alex Kurakin, Dan Boneh, Florian Tramr, Ian Goodfellow, Nicolas Papernot, and Patrick McDaniel. Ensemble adversarial training: Attacks and defenses. 2018. URL `https://arxiv.org/pdf/1705.07204.pdf`.

[26] Nicolas Papernot, Patrick D. McDaniel, Ian J. Goodfellow, Somesh Jha, Z. Berkay Celik, and Ananthram Swami. Practical black-box attacks against deep learning systems using adversarial examples. *CoRR*, abs/1602.02697, 2016. URL `http://arxiv.org/abs/1602.02697`.

[27] Nicolas Papernot, Patrick D. McDaniel, Arunesh Sinha, and Michael P. Wellman. Towards the science of security and privacy in machine learning. *CoRR*, abs/1611.03814, 2016. URL `http://arxiv.org/abs/1611.03814`.

[28] Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. In *NIPS Deep Learning and Representation Learning Workshop*, 2015. URL `http://arxiv.org/abs/1503.02531`.

[29] Nicolas Papernot, Patrick D. McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. *CoRR*, abs/1511.04508, 2015. URL `http://arxiv.org/abs/1511.04508`.

[30] Nicholas Carlini and David A. Wagner. Defensive distillation is not robust to adversarial examples. *CoRR*, abs/1607.04311, 2016. URL `http://arxiv.org/abs/1607.04311`.

[31] Chuan Guo, Mayank Rana, Moustapha Cissé, and Laurens van der Maaten. Countering adversarial images using input transformations. *CoRR*, abs/1711.00117, 2017. URL `http://arxiv.org/abs/1711.00117`.

[32] Weilin Xu, David Evans, and Yanjun Qi. Feature squeezing: Detecting adversarial examples in deep neural networks. *CoRR*, abs/1704.01155, 2017. URL `http://arxiv.org/abs/1704.01155`.

[33] Warren He, James Wei, Xinyun Chen, Nicholas Carlini, and Dawn Song. Adversarial example defenses: Ensembles of weak defenses are not strong. *CoRR*, abs/1706.04701, 2017. URL `http://arxiv.org/abs/1706.04701`.

[34] Jiajun Lu, Hussein Sibai, Evan Fabry, and David A. Forsyth. NO need to worry about adversarial examples in object detection in autonomous vehicles. *CoRR*, abs/1707.03501, 2017. URL `http://arxiv.org/abs/1707.03501`.

[35] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Computer Vision and Pattern Recognition (CVPR)*, 2015. URL `http://arxiv.org/abs/1409.4842`.

[36] Nicolas Papernot, Ian Goodfellow, Ryan Sheatsley, Reuben Feinman, and Patrick McDaniel. cleverhans v1.0.0: an adversarial machine learning library. *arXiv preprint arXiv:1610.00768*, 2016.