Stateful detection of black box adversarial attacks

Steven Chen

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2019-55
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2019/EECS-2019-55.html

May 17, 2019

Abstract

Stateful Detection of Black-Box Adversarial Attacks

by

Steven Chen

Master of Science in

University of California, Berkeley

Professor David Wagner, Chair

The problem of *adversarial examples*, evasion attacks on machine learning classifiers, has proven extremely difficult to solve. This is true even when, as is the case in many practical settings, the classifier is hosted as a remote service and so the adversary does not have direct access to the model parameters.

This paper argues that in such settings, defenders have a much larger space of actions than have been previously explored. Specifically, we deviate from the implicit assumption made by prior work that a defense must be a stateless function that operates on individual examples, and explore the possibility for stateful defenses.

To begin, we develop a defense designed to detect the process of adversarial example generation. By keeping a history of the past queries, a defender can try to identify when a sequence of queries appears to be for the purpose of generating an adversarial example. We then introduce *query blinding*, a new class of attacks designed to bypass defenses that rely on such a defense approach.

We believe that expanding the study of adversarial examples from stateless classifiers to stateful systems is not only more realistic for many black-box settings, but also gives the defender a much-needed advantage in responding to the adversary.

Stateful Detection of Black-Box Adversarial Attacks

by

Steven Chen

A thesis submitted in partial satisfaction of the

requirements for the degree of

Master of Science

in

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor David Wagner, Chair
Professor Dawn Song

Spring 2019

Dedicated to my parents and family, who without their love and support, none of this would be possible.

Contents

Acknowledgments

This work was conducted with the advising of Professor David Wagner, and PhD alumnus Nicholas Carlini. Thank you to both of them for their consistent guidance and all-around help with developing and framing this project, especially for Prof. Wagner's regular feedback on the project's direction and Nicholas' experiments with the adaptive autoencoder attack. I would also like to thank PhD candidates Michael McCoyd and Chawin Sitawarin for their comments and revisions of this manuscript.

Chapter 1

Introduction

Over the past few years, neural networks have driven advancement in a wide range of domains. Deep learning based methods have achieved state of the art performance in areas including, gameplaying AIs for Go and chess [34], machine translation between different languages [39], and classification and object detection for images [32]. Accordingly, neural networks are also increasingly used in safety-critical applications, where the reliable performance of these networks and their security against a malicious adversary is paramount. In some cases, such as a local image recognition system, the network and its parameters may be available to the adversary (the white-box case). However, when the classifier is hosted remotely (e.g., as a cloud service), only the output of the neural network is available to the adversary (the black-box case).

Worryingly, these neural networks have been shown to be highly vulnerable to *adversarial examples*: inputs crafted by an adversary to deliberately fool a machine learning classifier. Defending against adversarial examples has proven to be extremely difficult. Most defenses that have been published have been found to have significant flaws [8, 4], and even those few defenses that have thus far withstood validation offer only partial robustness [28].

Adversarial examples might not actually be problematic in practice, where models are often held private by companies and hosted on the cloud. However, surprisingly, adversarial examples can even be generated in a fully *black-box* threat model. Such an adversary in this threat model can only make queries of the model and receive the predicted classification label as output. While there certainly are domains where the adversary will have white-box access to a deployed neural network, in many production environments when neural networks are deployed, the user is only allowed to make queries of the classifier and observe the output. For example, services such as Clarifai [12] and Google Cloud Vision AI [18] offer image classification APIs where users can submit images and receive only the label of that image. Similarly, for spam classification, a feature offered by many email providers, a user cannot directly access the actual spam classifier, but only observe whether an email is classified as spam or not.

We study the problem of detecting the *generation* of adversarial examples, as opposed to trying to (statelessly) detect whether or not any *individual* input is malicious (which has

proven to be difficult [8]). To do this, we consider the task of detecting the *sequence* of queries made to the classifier when creating an adversarial example. The central hypothesis we evaluate is whether the sequence of queries used to generate a black-box adversarial example is distinguishable from the sequence of queries when under benign use.

Operating under this hypothesis, we propose a defense that relies on the specific observation that existing black-box attacks often make a sequence of queries, where each query is similar to the prior. In contrast, benign users rarely make queries for multiple nearly-identical copies of the same image. We train a similarity-detector neural network to identify such query patterns, and find that the existing state-of-the-art black-box adversarial example attack algorithms can be easily detected through this strategy. Our proposed strategy can trivially compose with any existing defense for defense-in-depth.

Then, we study adaptive attacks against our scheme, to understand whether an attacker who is aware of our detection strategy could evade it. We develop *query blinding*, a general strategy for attacking defenses which monitor the sequence of queries in order to detect adversarial example generation. Query blinding attacks pre-process each input with a *blinding function* before querying the classifier, so that (1) the pre-processed inputs match the benign data patterns, but (2) it is possible to deduce the classifier's output from the result of these queries. We show that our defense remains secure against query blinding.

Given the difficulty in defending or detecting attacks statelessly, we believe that this new research direction—stateful methods for detecting black-box attacks—presents renewed hope for defending against adversarial example attacks in the black-box threat model.

Chapter 2

Background & Problem Statement

2.1 Preliminaries

This paper studies evasion attacks on neural networks [6], commonly referred to as adversarial examples [35]. We briefly cover background which will be familiar to readers who have followed this line of work.

Neural Networks. A neural network is a function $f(\cdot)$ consisting of multiple layers. Each layer computes a weighted linear combination of the outputs of the previous layer, followed by a non-linearity. Because neural networks can have arbitrarily many layers, as well as various non-lineariaties, they can provably approximate arbitrarily complicated functions. The *weights* θ of a neural network refer to the parameters used in the weighted linear combinations. To be explicit we may write $f_\theta(\cdot)$ to refer to the network $f(\cdot)$ with weights θ. Most of the recent impressive results in machine learning have come from applying neural networks [34, 39, 32]. This paper focuses on *classification* neural networks, where some example x is processed by the neural network to return the predicted *label* $y = f(x)$ of the example.

Training Neural Networks. A neural network begins with randomly initialized weights θ. The process of training a neural network allows the network to iteratively learn better weights to solve the given task.

Neural networks are most often trained with stochastic gradient descent. Given a set of labeled training examples $\mathcal{X}$ with examples x_i and corresponding labels y_i, gradient descent attempts to solve the problem

$$\underset{\theta}{argmin} \underset{(x,y)\in\mathcal{X}}{E} \ell(f_\theta(x), y)$$

where $\ell(\cdot)$ measures the *loss*: intuitively, how "wrong" the prediction $f_\theta(x)$ is compared to the true label y. The process of stochastic gradient descent solves the above problem by

iteratively updating the weights

$$\theta \leftarrow \theta - \varepsilon \cdot \nabla_\theta \Big(\sum_{(x,y)\in B} \ell(f_\theta(x), y) \Big)$$

where ∇_θ is the gradient of the loss with respect to the weights θ; $B \subset \mathcal{X}$ is a randomly selected *mini-batch* of training examples drawn i.i.d. from $\mathcal{X}$; and ε is the *learning rate* which controls by how much the weights θ should be changed.

Adversarial Examples. To formalize the definition of adversarial examples and make their study well-defined, most existing work [17, 28, 4] defines an adversarial example as an input x', which is a slightly modified version of a naturally occurring example x, such that a neural network classifies them differently. Formally, an adversarial example x' satisfies two properties: (1) for some $d(\cdot)$, a distance metric, $d(x, x') < \varepsilon$, but (2) for the neural network, $f(x) \neq f(x')$. As long as ε is set to be small enough, the perturbation that is introduced should not change the actual true classification of the object in the image (e.g. a dog with small peturbations is still a dog).

Generating Adversarial Examples. The problem of generating adversarial examples can be formalized as a minimization problem

$$\delta^* = \underset{\delta}{argmax}\, \ell(f(x+\delta), y)$$

subject to the constraint that δ is small according to some metric.

White-box attacks to generate adversarial examples largely rely on using the same gradient-descent process used to train a neural network. Initially, we set $\delta_0 = 0$ and then update

$$\delta_{i+1} = \delta_i + \nabla_x \ell(f(x+\delta_i), y)$$

for some chosen loss function ℓ.

Black-box attacks, by definition, are unable to perform the above optimization because they are not able to compute the gradient of the loss. Instead, black-box attacks must perform gradient-free optimization. This paper considers two possible state-of-the-art black-box attacks: NES [21] and the BoundaryAttack [1]. While their implementations differ (significantly so), at a very high level they both rely on the same strategy. Starting from some initial perturbation δ_0, the attacks slowly query the classifier on a sequence of perturbations $f(x+\delta_i)$, each highly similar to the previous, with the objective of finding an input that is (a) misclassified and (b) introduces a small distortion.

A thorough understanding of these black-box attacks is not necessary yet; we defer a complete description to Section 6.

Problem Domain: Image Classification. Following most prior work on the space of adversarial examples [4], this paper studies the domain of image classification. Here, images are represented as $h \cdot w \cdot c$ dimensional vectors (with height h, width w, and c color channels) drawn from $[0,1]^{hwc}$.

2.2 Threat Model

As discussed earlier, we focus on detecting black-box attacks for crafting adversarial examples. In a black-box threat model, the adversary can query the model on any input and learn its classification, but the weights and parameters of the model are not released to users. We envision, for example, a platform that makes available a machine learning model as a service, where the model can be queried by a user after he/she creates an account, but cannot download the model itself. Under this threat model, we aim to increase the difficulty for attackers to craft adversarial examples. While an attacker can query the model any number of times in trying to generate an adversarial example, our goal is to detect such attacks before they are successful.

We focus on an account-oriented setting, where users must create an account before they can query the model. Attackers may be free to create as many accounts as they wish, but we assume there is some practical cost associated with creating each account (e.g., linking to a valid credit card or phone number, paying an account fee, etc.). In our scheme, the attacker's account can be cancelled as soon as an attack-in-progress is detected, requiring the attacker to create a new account at that point. A key metric for the effectiveness of our defense is the number of accounts that an attacker must create to successfully craft an adversarial example. Each time the attack is detected, the attacker must create a new account, so we measure this by counting the number of times the attack is detected before it is successful (number of detections); this determines the attacker's cost to defeat the system.

Notice that our goal of detecting when an adversarial attack is in progress over a sequence of queries to the model is different from detecting whether or not any individual input is adversarial (as in previous detection based defenses [8]). Thus, our scheme involves retaining history of prior queries and scanning this history to check for patterns that indicate if an attack is in progress. Such a defense is not feasible in the white-box threat model, where the defenders have no visibility into the attacker's offline computation.

We focus on the *hard-label* setting, where each query to the model returns only the categorical label assigned by the classifier, but not a numerical confidence score associated with it. Our approach would extend naturally to other settings, but as argued in prior work [6] we believe the hard-label setting is the most realistic black-box threat model.[1]

There are two broad types of black-box attacks in the literature: *query-based attacks*, which make a sequence of queries to the model, and *zero-query attacks*, which work entirely offline without interacting with the model. While significant prior work has been dedicated to constructing defenses against the latter [36], limited work studies defenses against query-based attacks. Our main contributions lie in defending against query-based attacks. Our approach, monitoring the sequence of queries against the classifier, by definition can not detect zero-query attacks.

[1] We have some evidence (see Appendix E) that solving the soft-label setting may be more challenging. We leave it to future work to study that problem area.

2.3 Query-Based Attacks

Our defense is motivated by the sequential nature of query-based black box adversarial attacks, such as NES [21] and the Boundary Attack [1]. Query attacks iteratively perturb a source example to slowly transform it into an adversarial example according to some policy, usually by estimating gradients or boundary proximity. This information is inferred by querying the current proposed adversarial example and points near the example.

If the defense considers these attack queries as a sequence, then successive queries are likely to be close together (by some distance metric), because (1) each iteration of the attack makes a small estimated-gradient or boundary-based step from the current proposed adversarial example to the next proposed example, and (2) since only labels are accessible, the attack requires querying a random sample of points near the current example (to approximate the actual gradient or decision boundaries of the model). Therefore, a scheme that tracks the sequence of queries made to a model should be able to detect an attack based on an anomalous pattern of suspiciously close queries.

As a concrete example, the label-only version of the NES attack [21] starts with an image that is already adversarial (it has a different class from the original image) and then sequentially takes "gradient" steps from the adversarial image towards the original image. Because obtaining a true gradient is not possible in the hard-label setting, the attack uses the gradient-free optimization method, NES, which estimates the gradient using the softmax scores of a random sample of points nearby the original example x. For this gradient estimation to be accurate, queries must necessarily be made within a small distance of each other, which creates a pattern that we will use to detect the NES attack.

2.4 Zero-Query Attacks

One of the most surprising properties of adversarial examples is their *transferability* [17]: given two different models (even trained on different datasets) for the same task, it turns out that adversarial examples generated on one will often transfer to the other. This observation motivated the earliest black-box attack algorithms: train a "surrogate model" [30] on the same task as the target model, perform a white-box attack on the surrogate model, and replay this generated adversarial example on the target model. This zero-query attack, while not 100% successful, is surprisingly effective.

As mentioned earlier, our approach cannot defend against transfer attacks and other zero-query attacks. However, others have proposed possible defenses to transfer attacks. Perhaps the best known example is Ensemble Adversarial Training (EAT) [36] which has been shown to be effective against zero-shot adversarial attacks.

The major limitation of these zero-query defenses (including EAT) is that they are not effective against query-based attacks. Fortunately, the prior work of defenses targeting the zero-query threat model perfectly complements our approach: we envision combining our defense (to detect query-based attacks) with an existing defense (to detect zero-query attacks).

In Section 8 we combine EAT with our defense to develop a complete defense to black-box adversarial examples.

Chapter 3

Our Scheme

We now introduce and explain our scheme to detect black box, query based, adversarial attacks by tracking the sequence of queries the attacker makes in the process of generating an adversarial example.

3.1 The Query Detection Defense

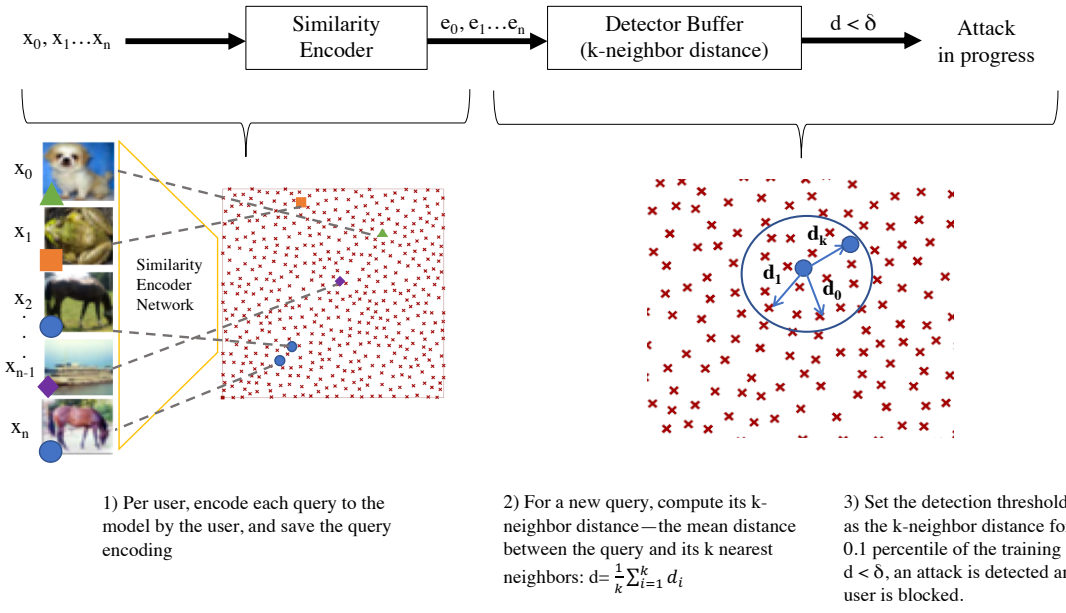

Figure 3.1: **Query Detection Defense:** The high-level process for detecting a query-based adversarial attack.

We propose a simple strategy to detect query-based attacks for generating adversarial examples. At a high level, our defense is applied as an access monitor on top of an existing classifier. Our detector records all queries to the classifier and stores them in a temporary history buffer. For each new query, the detector computes the number of "nearby" examples in this temporary history buffer. If we determine there are too many nearby examples, we report this as part of an attack sequence and take appropriate action (e.g., block this user's account).

In more detail, for each user, we save every query from that user for a bounded duration (this period can be tuned according to the defender's resources, for example either a fixed amount of time or a fixed number of queries). Then, for each new query the system receives, we compute its *k-nearest-neighbor distance* to the previously seen examples—the mean pairwise distance between the query and its k nearest neighbors among the previously saved queries (i.e., for each of the k nearest neighbors, we compute the distance between the neighbor and query, then take the mean over these k distances). To measure the distance between queries, we first encode the queries using a deep similarity encoder [5], that maps perceptually similar images to nearby points in a reduced dimensional space, and then use the ℓ_2 distance in this encoded space.

If the mean distance falls below some chosen threshold, then we flag this example as a potential attempt at generating an adversarial example. We choose the detection threshold so that benign use of the classifier is not flagged. In particular, we set the threshold so that if the entire training set were to be randomly streamed as queries, 0.1% of the training set would be flagged as attacks (i.e. the false positive rate would be 0.1%). [1] After an attack is detected, the buffer containing the previously saved queries for that user can then be cleared. Moreover, in response to the attempted attack, the user may then be banned from the service either immediately, or after a random number of subsequent queries, in order to reduce the attacker's knowledge of when exactly their attack was detected. A diagram of our scheme is shown in Figure 3.1.

3.2 Similarity Encoder

A key question in the design of our method is the metric to use for the k-nearest-neighbor search. Naively, we might imagine choosing a simple metric—for example, the ℓ_2 distance between two images. However, using such a simple method has two drawbacks:

1. Most importantly, simple metrics, such as ℓ_2, may not accurately capture distance in adversarial situations and are too easy for an attacker to evade. A small rotation or translation in pixel space can cause dramatic changes according to ℓ_2 norm, which experimentally we find allows an adversary to evade detection.

[1] In practice, a lower false positive rate may be necessary. However, some existing defense research for detecting adversarial examples sets the false positive rate at approximately 5% [41]. Our value is thus 50× lower than prior work.

2. Additionally, computing ℓ_2 distance requires storing an entire copy of every queried image. This could potentially pose a significant cost to the hosting service from both the storage costs, as well as the potential privacy risks that come with storing user queries for longer than strictly necessary.

To increase the difficulty of such circumvention techniques, we use perceptual similarity as the starting point for our distance metric, as by construction, adversarial attacks intend to generate examples perceptually similar to the original image. To measure the perceptual similarity of two images, we train a deep neural network to encode images into a lower-dimensional space of dimension d, such that similar images are mapped to similar points in the *encoded space*. For example, for a given picture of a dog, after rotating or translating the image slightly, the perceptual content of the image is still the same (i.e., the same dog), and we train the encoder so that both of these images have similar d-dimensional representations.

This construction resolves both of the difficulties identified earlier. By design, small modifications to an image are less likely to cause dramatic increases in encoded-space ℓ_2 distance. Further, because the encoded space is much smaller than the total image size, this allows us to save on storage costs.

Encoder Setup & Training. We represent the encoder $E(\cdot)$ as a neural network mapping images $x \in R^{h \cdot w \cdot c}$ to an encoded space $e \in M$ of dimension M. As described, the objective of this encoder is to map visually similar inputs $x, \tilde{x}$ to encodings $e = E(x)$, $\tilde{e} = E(\tilde{x})$ that are similar under ℓ_2 distance, so that $e - \tilde{e}$ is small.

To achieve this we train the similarity encoder neural network with a *contrastive* loss function [5]. Specifically, we consider two pairs of images. Pair 1 consists of a training set image x_i, and an image perceptually similar to x_i: a positive, x_p. Pair 2 consists of a different training image x_j with an image *not* perceptually similar to x_j: a negative, x_n. We then define the loss for their encodings $(e_i, e_p), (e_j, e_n)$ as the contrastive loss function

$$L(x_i, x_p, x_j, x_n) = \|e_i - e_p\|_2^2 + \max\left(0, m^2 - \|e_j - e_n\|_2^2\right).$$

The first term encourages similar encodings for positives and the second term encourages *different* encodings for negatives by penalizing encodings less than a certain margin m apart for negatives.

3.3 Experimental Setup

We evaluate our defense on the CIFAR-10 dataset: a collection of 60,000 low-resolution (32×32) color images drawn from 10 classes. Because ach pixel is a color value between 0 and 1, each image is drawn from $[0, 1]^{32 \times 32 \times 3}$. We choose this dataset for three reasons. First, CIFAR-10 is the most popular dataset for studying adversarial examples [4]. Second, defenses on ImageNet have thus far proven to be far beyond our current capabilities [13], and no propose neuradl network defense remains robust to attack. Finally, CIFAR-10 is

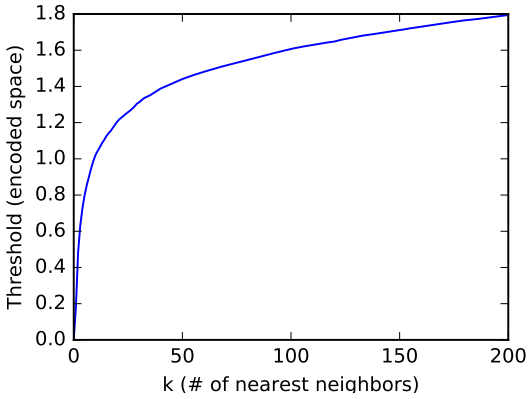

Figure 3.2: The mean k-Neighbor distance (in encoded space) of the 0.1% percentile of the CIFAR-10 training set as a function of k. We select the threshold k so that it is large enough to support a wide margin, but not so large as to be computationally prohibitive.

significantly less computationally expensive than ImageNet, allowing for us to perform a wide range of experiments.

We train a ResNet classifier [20] on the CIFAR-10 dataset for 100 epochs with Adam [25] on a 1080 Ti GPU with Keras [11] and TensorFlow [2], achieving 92% test accuracy.

3.4 Encoder Training and Threshold Selection

To train the similarity encoder for the scheme, we follow the advice of prior work [5] and initialize our encoder with the same architecture and weights as a network trained to classify the desired images. For CIFAR-10, we first train a three layer CNN (architecture given in the appendix) for 100 epochs using data augmentation and reach a validation accuracy of 76%. Then the similarity encoder is constructed by replacing the logits layer of the network with a dense layer of 256 units, to produce an encoding of dimension $d = 256$. A margin of $m = \sqrt{10}$ was found experimentally to result in the best encodings. A more complex architecture such as a ResNet [20] might yield even better results, but this simpler architecture was sufficiently effective.

The similarity network was trained to minimize the contrastive loss function described in Section 3.2, with a learning rate $\alpha = 1e-4$, momentum $\mu = 0.9$, and a batch size of $b = 32$. To generate a batch of positive pairs for training, b images were randomly selected from the training set; then, a random image transformation (that should retain the perceptual content of the image) was selected and applied to each of the b images (similar to traditional data augmentation used for network training). For negative pairs, b pairs ($2b$ images total) of different images were randomly selected without replacement. The transformations used are enumerated in Section 5.1.

Threshold selection. The choice of k for our scheme is important, since the detection threshold is determined by the k-neighbor distance of the 0.1 percentile of the training set. The value for k should be selected such that the detection threshold is maximized (increasing the separation between suspicious and benign queries), while keeping k relatively low (to reduce the computational cost of calculating the k-neighbor distance). However, also important is that k is now the minimum number of queries before our defense could possibly flag a possible attack: and so a smaller k should be preferred whenever possible. We set k by exploring the threshold needed to ensure a 0.1% false positive rate as a function of k.

As seen in Figure 3.2, this distance increases sharply until $k = 50$, where the distance begins to plateau and marginally continues to increase as k increases. Therefore, we select $k = 50$ for the rest of this paper.

Chapter 4

Non-Adaptive Evaluation

Having described our defense proposal, we begin by demonstrating that it has at least some potential utility: it effectively detects existing (unmodified) black-box query attacks. While there are many black-box (hard-label) attacks, they fall roughly into two categories:

- **Gradient estimation** attacks at their core operate like standard white-box gradient-based attacks (as described in Section 2.1). However, because they do not have access to the gradient, these types of attacks *estimate* the gradient through repeatedly querying the model.
- **Boundary following** attacks, in contrast, first identify the decision boundary of the neural network, at a potentially far-away point, and then take steps following the boundary to locate the nearest point on the boundary (i.e., an adversarial example) to the given input.

We evaluate against one attack from each category as representative examples.

4.1 Attack Setup

For each attack studied, we use the *targeted* variant, where the adversary generates an adversarial example chosen so that the resulting adversarial example x is classified as a target class t and is within a distance ϵ of an original image x. The original image and target class are chosen randomly. We call an attack successful if the ℓ_∞ distortion is below $\varepsilon = 0.05$. While most white-box work on CIFAR-10 considers the smaller distortion bound of $\varepsilon = 0.031 \approx 8/255$, we choose this slightly larger distortion because black-box attacks are known to be more difficult to generate and so we give the adversary slightly more power to compensate.

NES [21] is one of the two most prominent gradient-estimation attacks (along with SPSA [38]). It estimates the gradient at a point by averaging the confidence scores of randomly sampled nearby points, and then uses projected gradient descent to perturb an image of

Attack	Success Rate	Num. Queries	Detections
NES	100%	325,200±153,300	6,377
Boundary	100%	14,720±8,923	288

Table 4.1: **Success rate of unmodified attacks on a neural network protected with our scheme.** While attacks succeed with 100% probability, the attacks trigger between hundreds to thousands of detections.

the target class until it is sufficiently close to the original image. In the hard label case, the confidence score for a point is approximated by taking a Monte Carlo sample of nearby points, and then computing the score for a class as the fraction of nearby points with that class.

The Boundary Attack [1] was the first attack to propose following the decision boundary to generate black-box adversarial examples. Since its publication, there have been multiple proposals to improve this attack [24, 22]. We still evaluate our defense on the vanilla boundary attack; the other attacks are more query efficient, but at their core still perform the same operation. To compare directly with the NES attack, we use ℓ_∞ distance with the Boundary attack (instead of the usual ℓ_2 distance).

4.2 Results

We run the default, unmodified implementations of each attack against our scheme and find that they can be detected. The results are presented in Table 4.1. (Attack specific parameters for NES are given in 6.1). An attack is considered successful if an adversarial example is found within an ℓ_∞ distortion of $\epsilon = 0.05$ from the original image, and of the target class is produced. The attack also terminates as soon as it finds such an example. Each attack does eventually succeed at a high rate, but is detected frequently with at least 200 detections on average. Thus, an attacker would need to create at least 200 accounts in order to generate a single adversarial example with these attacks. This demonstrates that our query sequence based scheme can reliably detect existing black box attacks.

Chapter 5

Query Blinding

While showing that our proposed defense can detect existing attacks is a useful first step, it is not sufficient for a complete evaluation. We must also evaluate whether our defense can detect future attacks. Doing this requires developing *adaptive attacks* specifically designed to bypass the defense proposal.

Thus, we introduce *query blinding*[1]: a general strategy which can be used to hide the query sequence from the defender. At its core, the objective of a query blinding attack is to learn the value of $f(x)$, for some specific x, without actually querying the point x. We define two functions: a randomized blinding function $b(x; r) = \{x'_0, x'_1, \ldots, x'_n\}$ that maps from one example to a set of modified examples so that $x'_i - x \geq \varepsilon$, and a revealing function $r(f(x'_0), f(x'_1), \ldots, f(x'_n)) = y \approx f(x)$ that approximates what $f(x)$ would return given the classifier outputs. In this paper for simplicity we restrict ourselves to the case where $b(x) = 1$ for simplicity. In the appendix we give an example of a more sophisticated blinding function that deviates from this simplifying assumption.

5.1 Image Transformations

Image processing transformations, such as image translation and brightness adjustment, are natural and readily available blinding functions. Let x be the image that an attacker would like to query the model for, $f(x)$ be the model's output for the query, and $T_c(x; r)$ be a randomized image processing transform (e.g., by rotating it or shifting it by a random amount c). We would then set $b(x; r) = \{T_c(x; r)\}$. Because the purpose of our blinding function is to fool the query-detector by transforming a sequence of queries which are pairwise similar to a sequence of queries which are not, we would like the ℓ_2 distortion between the original image and the transformed image to be large. For example, for a 3 x 32 x 32 CIFAR-10 image, adjusting the brightness (i.e. individual pixel values) by a factor of just $c = 0.001$ increases the ℓ_2 distortion by $0.001(3)(32)(32) = 3.072$. However, despite increasing distortion, these transformations still retain the primary content of the image,

[1]We call this *query blinding* because of its similarities to *blind signatures* [10].

and a model of high accuracy/performance should produce relatively similar outputs for the original and transformed images, so the corresponding revealing function for an image processing transformation is simply $r(f(x')) = f(x')$. This paper considers eight different possible transformations:

- *Uniform Noise*: add uniform noise to the image, where the noise is drawn from a uniform distribution $c \sim U(-r, r)$.
- *Translate*: translate the image horizontally and vertically by c_h and c_v fractional pixels, where c_h and c_v are sampled randomly from a uniform distribution $c_h \sim U(-r, r)$ and $c_v \sim U(-r, r)$, using bilinear interpolation and filling in empty space with zeroes.
- *Rotate*: rotate the image $c\pi$ radians, where c is sampled randomly from a uniform distribution $c \sim U(-r, r)$, using bilinear interpolation and filling in empty space with zeroes.
- *Pixel-wise Scale*: scale each pixel by the same factor c, where c is sampled randomly from a uniform distribution $c \sim U(1 - r, 1 + r)$
- *Crop and Resize*: crop the image to box coordinates of $[c, c, 1 - c, 1 - c]$ and then resize image to original size (using bilinear interpolation), where c is sampled randomly from a uniform distribution $c \sim U(0, r)$ and box coordinates of $[0, 0, 1, 1]$ represent the uncropped image.
- *Brightness*: adjust the brightness of the image by adding to each pixel the same constant c, where c is sampled randomly from a uniform distribution $c \sim U(-r, r)$.
- *Contrast*: adjust the contrast of the image by a random contrast factor c, where c is sampled randomly from a uniform distribution $c \sim U(r, 1)$.
- *Gaussian Noise*: add Gaussian noise to the image, where the noise is drawn from a Gaussian distribution $c \sim \mathcal{N}(0, I) * r$.

5.2 Auto-Encoder Attack

While simple image transformations are one possible method for performing query blinding, it is certainly not the optimal strategy. We now develop a strategy for *learning* a blinding function.

Our proposed approach directly trains a type of *auto-encoder* neural network $\alpha(x)$. Typical auto-encoders are trained with the property that $\alpha(x) \approx x$. Instead, given an input image x, we train a randomized auto-encoder $\alpha(x; r)$ to satisfy two properties: (1) $\alpha(x; r_1) - \alpha(x; r_2)|_2$ is large, but (2) $f(\alpha(x; r)) \approx f(x)$. Satisfying property (1) ensures that the augmented image will evade detection by the encoder, while property (2) ensures that the actual classification of the image will remain unchanged.

Specifically, we train the autoencoder to minimize the loss

$$\ell(x) = H\Big(f(\alpha(x,r)), f(x)\Big) - c \cdot \min\Big(\alpha(x;r_1) - \alpha(x;r_2)_2^2, d^2\Big)$$

where $H(\cdot)$ is the cross-entropy loss between the two distributions, c is a constant that controls the relative importance of the two loss terms, and d is a constant that sets the maximum desired ℓ_2 distance between transformed examples.

We train the autoencoder with stochastic gradient descent for 10 epochs on the CIFAR-10 training data. In order to ensure that we are not "cheating" by training on the exact function $f(\cdot)$ which we will be attacking, we train a new classification neural network $f'(\cdot)$ on 10% of the CIFAR-10 training data. In practice, we set $c = 1$.

To determine the threshold d, we try values between 2 and 20 and pick the one that is most effective at fooling the detector. We found that in practice $d = 10$ is well-balanced between being big enough so the detector is fooled, but not so big that $f(\alpha(x;r))$ is substantially different from $f(x)$.

5.3 Increasing Attack Diversity

While, on average, the above query-blinding attacks generate images with an ℓ_2 difference of some distance d between them, in the worst case (if we are unlucky with the choice of randomness) it may be possible that we have blinded images which are nearly identical. This is true especially when we generate a large number of transformations $x_i = b(x, r_i)$. By the birthday paradox, we should expect that as we generate increasingly many images, at least some fraction will be similar to each other.

For simplicity of analysis, assume for the moment that our defense relied exclusively on the ℓ_2 distance between images being less than some threshold τ to detect attacks. After we have made queries $q_0, q_1, \ldots, q_{j-1}$ we could now check whether or not making the query $q_j = b(x; r_j)$ would be detected as an attack against our own history. If it would be, we can simply re-sample a new value $r_{j'}$ and generate a new candidate query $q_j = b(x; r_{j'})$ until we obtain a sample that is above the detection threshold.

While this greedy approach is simple, we could do better if we knew *a priori* how many queries we would need to make of the target classifier. Begin by generating a large number of candidates $x_i = b(x, r_i)$. We construct a graph G where each candidate x_i is a node, and we connect node i to node j if the distance between x_i and x_j is less than the threshold τ (i.e., querying both would result in a detection. Formally, $G = (V, E)$ where $V = 1, \ldots N$ and $E = \{(i, j) : x_i - x_j < \tau\}$. Identifying the maximum subset of examples that would not cause a detection by our hypothetical ℓ_2-based detector then reduces to finding the maximum independent set of this graph G. While this problem in general is NP-Hard and NP-Hard to approximate, we find that simple approximation algorithms identify sub-graphs with a cardinally on average twice as large as our initial greedy querying approach.

While we in practice use an encoder and do not rely directly on ℓ_2 distance, we find that this approach still reduces the number of detections with the encoder.

Chapter 6

Adaptive Attack Evaluation

Given that our proposed defense effectively prevents existing black-box attacks, we now ask the question: can it prevent more sophisticated attacks designed to bypass this defense? We find that while it is possible to degrade the effectiveness of the defense, we can not defeat it completely.

6.1 The NES Attack

We first elaborate on the NES attack [21] in more detail. The NES attack generates an adversarial example by starting with an image of the target class (i.e. that is already adversarial) and then using projected gradient descent to perturb the example until it's within ϵ of the original image, while still classified as the target class. The gradient at a particular point, x, is approximated effectively as a finite differences estimate on a random Gaussian basis. Full details can be found in [21], but simplified, the gradient is estimated by: (1) sampling n instances of Gaussian noise from $\delta \sim \mathcal{N}(0, 1)$ and adding them each to x as $\theta = x + \sigma\delta$ to generate n basis points (2) for each of the n basis points, estimating the confidence scores at a point, θ, by querying the labels for s points around the point taken from a sampling ball of l_∞ radius μ (and computing the proportion of each label as that class's score), (3) averaging the estimated confidence scores while scaling by some σ to reduce variance.

The attack-specific parameters used for NES were the default parameters suggested in [21], and are as follows: $\sigma = 0.001$, $n = 4$, $s = 50$, $\mu = 0.001$, and learning rate $= 0.01$.

Parameter Tweaking

Aware of the scheme, an attacker may attempt to modify the NES attack in order to increase the ℓ_2 distance between queries. This would be sufficient to subvert the scheme if ℓ_2 is used as the distance metric. Even for a similarity based distance metric, since the similarity encoder is not available to the attacker, increasing ℓ_2 distance instead would be a natural

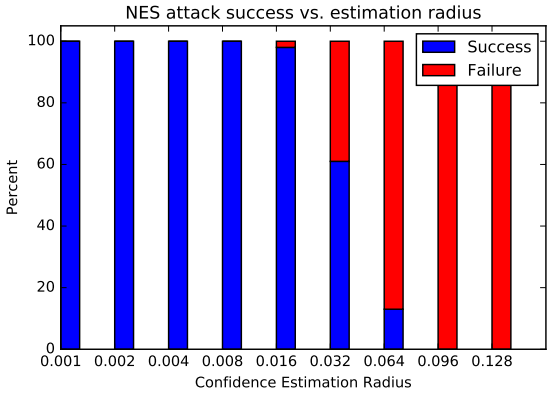

Figure 6.1: NES Attack Success Rate over increasing μ. For each value of sampling radius, μ, the NES attack was run 100 times, with the attack succeeding if an adversarial example (of a randomly chosen target and within $\epsilon = 0.05$) was found within 5 million queries, and failing otherwise.

proxy to attempt to circumvent detection. We now explore two natural modifications to the NES attack that an attacker could make.

First, the attacker could tweak different NES attack parameters to increase the ℓ_2 distance between queries. The attacker could increase μ, the radius of the sampling ball used when sampling points to estimate the confidence score for an image. The original version of the NES attack uses a radius of $\mu = 0.001$, but as shown in Figure 6.1, radii up to $\mu = 0.064$ may be used before the attack drastically fails. For $\mu = 0.064$, $k = 50$, and using ℓ_2, the k-nearest-neighbor distance between an image and k sampled points is on average equal to 2.32 compared to just 0.032 for $\mu = 0.001$.

By similar logic to increasing μ, the attacker could also increase σ such that the sampled Gaussian basis points are further apart. Additionally, the attacker could decrease s such that less queries are guaranteed to be within an l_∞ distance of μ at each iteration. Experimentally, we found that the success of the NES attack with increased σ and/or decreased s are both sensitive to the confidence estimation procedure (i.e. more accurate estimation procedures allow for higher σ and lower s). The effect of modifying these parameters is investigated further in the appendix.

Query Blinding

The second natural modification is for an attacker to transform their queries to have greater ℓ_2 distortion, as described previously. Each query is transformed by an independent random amount, as opposed to transforming each query by the same amount of distortion, because the sampling procedure for estimating confidence scores in NES requires a random sam-

ple of different points. Transforming by a random amount also increases the likelihood of consecutive queries being further from each other.

Accordingly, we modify the confidence score estimation procedure (step 2 from the previous attack description) to sample s points using the different transformations listed in Section 5.1 (instead of sampling just from a l_∞ ball of uniform radius). The parameters for each transformation were chosen such that the expected ℓ_2 distortion from each transformation was equal to 2.32 (the expected ℓ_2 distortion after increasing the uniform sampling radius to $\mu = 0.064$, the largest value/potential distortion where the original NES attack still succeeds), and are given in the appendix. This set of transformations and their respective parameters were also then used when training the similarity encoder, as described in Section 3.2.

	Transform	Success Rate	Num. of Queries	ℓ_2 Detections	Sim. Detections
Low Distortion	Uniform Noise	1%	15,700±0	308	308
	Translate	4%	6,710±400	131	131
	Rotate	7%	10,200±684	199	198
	Scale	27%	12,600±882	246	246
	Crop and Resize	7%	6,130±247	119	119
	Brightness	55%	13,500±780	263	264
	Contrast	23%	11,200±777	219	219
	Gaussian Noise	0%	n/a	n/a	n/a
High	Brightness	43%	24,500±2,630	60	481
	Scale	42%	25,700±3,000	88	504
	Contrast	37%	19,100±4,240	85	375
	Auto-encoder	76%	13,400±8,400	12	97

Table 6.1: **NES attack success rate for different blinding functions.** Simple transformations, such as rotation or contrast adjustment, are still detected by our encoder defense, even when the distortion introduced is large. The autoencoder attack is much more effective (especially if we were using a pure ℓ_2-based detector) but still is detected by the encoder defense.

We ran the NES attack, each run with a different transformation to preprocess queries during confidence score estimation, against our scheme, and the results are shown in Table 6.1. For some of these transformations, like uniform and Gaussian noise, the NES attack fails completely. However, for the brightness and pixel-scale transformations, the NES attack remains reasonably successful at even better rates than the original confidence estimation procedure of uniform noise. This suggests that estimating the confidence score for an image/point may be more accurate with certain image transformations than others, and that some transformations more significantly affect the network output for an image. For all transformations, however, the average number of detections for each successful attack is at

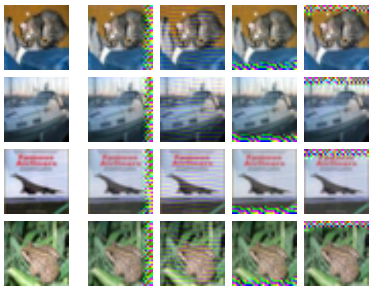

Figure 6.2: For each image in the test set (left) we show four possible images transformed by our autoencoder (right) that have the same classification label but have high ℓ_2 distance from each other.

least one hundred, so even after some adaption by the attacker, the scheme still detects these attacks frequently. The exact attacker cost corresponding to this number of detections is quantified and explained further in Section 7.

For this level of transformation distortion, using the similarity encoding as the distance metric over ℓ_2 shows a marginal increase in the number of detections. This is understandable, as for $k = 50$, the ℓ_2 detection threshold is $\delta = 5.093$ which is greater than the transformation distortion of 2.32 (which was selected based on the largest value for μ where the original NES attack was still successful). Accordingly, to strengthen the attack against our scheme, we selected the transformations with the highest success rates, and further modified their parameters to increase the transformation distortion to an ℓ_2 distance greater than the ℓ_2 based detection threshold. Specifically, we increased σ to be $\sigma = 0.01$ (the highest value found experimentally such that the attack still succeeds reasonably often), and increased the distortion parameter for each transformation, such that the expected distortion was equal to 5.40 and greater than the detection threshold $\delta = 5.309$ (the higher distortion parameters can be found in the appendix). To also minimize detections, we reduce s to $s = 2$. We then ran these higher distortion attacks again against our scheme, and the results are also shown in Table 6.1.

Against these higher distortion transformations, the similarity encoding distance metric more frequently detects attacks compared to the ℓ_2 based distance metric, resulting in 6-8x more detections. As explained later in Section 7, this multiplicative increase in detections can cause a significant increase in the cost to the attacker of carrying out a *single* attack. This demonstrates that a similarity based metric is more robust to the adaption of an attacker using image transformations to blind their queries, and less directly circumvented compared to using ℓ_2 distance. We also note that the number of similarity-based detections is larger than for the lower distortion transformations shown in Table 6.1 for the same $s = 2$, but observe that this is because of the attack taking more queries on average to succeed (due to less precise gradient estimates) and thus resulting in more opportunities for detection.

When we apply the auto-encoder attack, we find that it is significantly more effective than the simple transformations. (We give examples of the auto-encoded images which are

sent to the classifier in Figure 6.2.) In particular, because the neural network was trained so that it maps images to have high ℓ_2 distortion from each other, an ℓ_2-based detector (directly on image pixel-space) is almost completely ineffective against this attack. However, the encoder-space detection defense remains effective.

6.2 The Boundary Attack

The Boundary Attack [1] is a gradient-free attack that starts with an image of the target class and then makes steps alternating between moving the image along the decision boundary (while remaining adversarial) and steps which move towards the original image. In more detail, the boundary attack alternates between the two following operations:

- **Inwards step.** Given the current proposed adversarial example x'_i, we take a small step ε in the direction of the original image x and let $x'_{i+1} = x'_i + \varepsilon \cdot (x - x'_i)$.
- **Orthogonal (boundary-following) step.** Instead of steeping toward the original image, take the current adversarial example x'_i and take a step along a random orthogonal direction r subject to the constraint that $f(x'_i + r) = f(x'_i)$.

Parameter Tweaking.

Starting with the original Boundary Attack, an attacker may also use different variants of the attack in an attempt to avoid or minimize detections by our scheme. First, instead of sampling random directions from a Gaussian distribution, as in the original attack, an attacker may sample from a distribution of Perlin noise patterns. This substitution was shown to achieve higher success rates with a limited amount of queries (less than 15,000 queries) [7], and could decrease the number of queries required by the attack such that the number of detections is significantly small. However, we find that this strategy is not effective. Additionally, increasing the step sizes to be large enough so that they are not detected by our similarity encoder prevents the attack from converging.

Query Blinding.

Alternatively, the use of blinding transformations can also be applied to this attack. To evaluate this for an attacker, we take the best performing transformation with the NES attack (brightness) and use it to preprocess all queries made by the original Boundary Attack.

The results of running these variants of the Boundary attack are shown in Table 6.2. We allow each attack trial to make up to 200,000 queries, which is the equivalent number of queries made by the best NES attack variants. Compared to the normal Boundary attack, the Perlin distribution variant does not perform better, likely due to the observed decreasing utility of the Perlin noise at these higher query numbers [7]. Preprocessing queries with the brightness transform also does not decrease the number of detections while significantly

decreasing the Boundary attack's success rate. We found that this was the case because the Boundary attack adjusts its step sizes according to the local geometry of the boundary (estimated through queries), and considers the attack to have converged when the step sizes are sufficiently small. The preprocessed queries, though at a further distance apart, would cause the attack to misapproximate the local boundary geometry around the working adversarial example and reduce its step sizes too significantly, before the example was within the maximum amount of distortion (i.e. converge prematurely).

Nonetheless, even with the most efficient variant being the normal Boundary attack, our defense still detects this attack with high frequency of at least 200 detections and one detection per 50-55 queries (i.e. almost every k queries), on average. Therefore, to be detected a small number of times (say less than 10), a future variant of the Boundary attack would need to succeed consistently within less than 500 queries on average.

Attack	Success Rate	Num. of Queries	Similarity Detections
Boundary (Normal)	100%	14,700±892	288
Boundary (Perlin)	40%	31,100±976	609
Boundary (Brightness)	2%	24,100±1,470	473
Boundary (Auto-encoder)	61%	34,200 ± 8,000	240
Query Surrogate (Untargeted)	14%	8,192±0	82

Table 6.2: **Attack success and detection rate of the Boundary Attack and the Surrogate Attack.** Applying simple transforms (such as with perlin noise) performs *worse* than using transformation; applying an autoencoder-based blinding function slightly reduces detection rate but does not succeed as often.

6.3 Hybrid Query Based Surrogate Attack

Query based attacks iteratively modify a working adversarial example, so they naturally require queries that are nearby each other and made sequentially. To use queries that can be made in any order and are a large distance apart (i.e. detection-minimizing queries), we design a simple attack that per attack instance, trains a surrogate model (similar to zero-query attacks) over a set of well-distanced queries. Specifically, for an image x, we randomly sample n points, $x_{far} = x + \delta,\ \delta \sim U(D, 1)$, where D is large enough that in expectation a sample would not be detected, and query the sampled points for their labels. To ensure that some training points of x's class are present, we also sample m points, $x_{near} = x + \delta,\ \delta \sim U(0, d)$, where d is small, but do *not* query these points and assume their labels are of the same class as x, which is normally the case unless x is very close to a decision boundary. We then train a surrogate model on these $n + m$ points, and then run a white box attack, the Carlini-Wagner attack [9] with high confidence ($\kappa = 100$), in order

to generate an adversarial example that hopefully transfers. In our scheme, the similarity encoder is not made public, so the attacker would not know the exact value of D, but we consider this optimal setting of information for the attacker/worst case for the defender as a baseline to evaluate this attack.

In practice, we found that targeted version of this attack had difficulty succeeding, as it was difficult to consistently sample queries both of the target class and sufficiently distant from the original image. Therefore, for the evaluation of this attack, we relaxed the attack to target the class for which the surrogate training set had the most instances of and was not the same as the original image's class. The results of running the attack are shown in Table 6.2, and attack-specific parameter values are given in the appendix. As a simple, naive approach, the attack is still able to succeed at a non-trivial rate, while having significantly less, 3-8x, detections compared to the other attacks. This indicates that there is potential for constructing query based attacks that do not rely on sequences of nearby points, but such attacks may still be reasonably detected. We leave it to future work that could be done on improving such attacks (e.g. informed instead of random sampling) against this defense.

Chapter 7

Economics of Performing an Attack

Assuming our defense was in place, what would the economics look like for an attacker who wished to fool the defense?

Evading the detector First, let us consider the case where an attacker has perfect information for whether or not each query to the detection scheme will be detected as part of an attack sequence. Further assume the attacker knew the exact number of queries k which could be issued before a detection would trigger. Recall that, typically, this is not the case: normally, the encoder is not made public in our scheme. How should an adversary proceed to optimally generate an adversarial example?

Recall that we proposed two defense setups. In one, the defender holds a time-bounded history of prior queries, and in the other, the defender holds a fixed number of queries (e.g., in a circular buffer).

Case 1: time-bounded buffer. Consider the first case where the buffer holds queries until a fixed amount of time has passed. If the buffer has a duration of t hours, the attacker can only make $k - 1$ queries, every t hours or have their attack be detected. Therefore, an attack that would otherwise have had d detections would take an attacker dt hours to execute. Because the lowest average number of detections per targeted attack is at least 97 detections (for the NES attack with Auto-encoder blinding in Table 2), the average amount of time to execute a single attack, with a buffer duration of 1 hour, would be 97 hours or a little over 4 days, which is a significant amount of time for an attacker to expect on average to generate a single targeted adversarial image without being detected.

In practice, the defender can set the buffer duration according to their resource capacity. For example, Google's Cloud Vision API currently has a rate limit of 1800 queries per minute per user. If we were using a ℓ_2 encoder, we would have to store every queried image completely. Because CIFAR-10 images are 32 x 32 x 3 bytes, a buffer duration of 1 hour would equate up to at most (without any image compression) $1800 \cdot 60 \cdot 32 \cdot 32 \cdot 3 = 0.33GB$ per user for a maximum of one hour. This is a small storage cost, considering that Google's cloud storage is priced at 0.026$/GB/month per user.

However, with our similarity encoder, the cost is even smaller. Our similarity encoder maps each input to a 256-dimensional output. At this size, storing all 1800 examples (with 16-bit floating point values) would be under 55 MB of storage, over 6× smaller.

Case 2: query-bounded buffer. Now we consider the case where the buffer holds a limited number of queries for an unbounded (or much longer) amount of time. Assume the buffer held at most N queries. After the attacker has issued $k-1$ queries, the attacker must now flush the buffer to remove these queries before continuing with their attack. In the best case (for the attacker—the worst case for the defender) the attacker may only have to flush just one query from the buffer before being allowed to continue the attack. Thus, in order to execute an attack that otherwise would have had d detections would take an attacker $d(N-k)$ total queries.

For example, Google's Cloud Vision API currently costs $1.50 USD per 1000 queries. If the buffer held just $N = 10^4$ examples, for $N >> k$ (as is true in our case), and again the lowest number of detections on average for a targeted attack being $97 \approx 100$ detections, an attack would cost $(\$1.50/1000\,queries)\cdot(d\cdot N) = (\$1.50/1000\,queries)\cdot(100\cdot 10000) = \$1,500$, a sizeable amount of money for an adversary to pay for generating a single adversarial image, compared to a baseline cost of $19.50 USD.

Attacking the similarity encoder Throughout the paper we have assumed the adversary is not be able to perform an *extraction attack* in order to obtain a copy of the similarity encoder. If an adversary could obtain access to the similarity encoder, or extract the detection threshold, it would be possible to generate an almost-unbounded number of queries, none of which would ever be detected as part of an adversarial sequence of queries.

We believe that this attack is extremely unlikely to happen. Economically, it would be significantly more expensive than paying the cost for detections during a typical attack construction. Below we justify why we believe this to be the case.

At minimum to accurately extract a network with W parameters, at least W samples or datapoints are needed, and signficantly moreso when only the hard labels are outputted from the network rather than logits or scores [37]. In the case of extracting the similarity encoder, each sample/datapoint is not a query but a detection. Since a detection results in an account ban, at least W accounts are then required to extract the similarity encoder. Such a large W provides a significant upfront cost of accounts to the attacker, and although the attacker would then have effective access to the encoder, the defender could respond by simply training another or multiple autoencoders with a different architecture or different training parameters (e.g. different margin m) and periodically rotating between them.

To only extract the detection threshold, an attacker could purposefully construct queries close to each other in order to cause a detection, and then perform a line or binary search on the ℓ_2 distance required for detection (the ℓ_2 distance would be the actual threshold for an l_2 based detector, or an l_2 proxy of the threshold for a similarity detector). After extracting the threshold, the attacker may then be able to craft or order their queries in such a way

to explicitly pass the threshold. This attack approach could be explored in future work, but a candidate modification to our scheme to respond to such an attack is to randomize or delay when a user is banned after being detected. Instead of immediately banning the user/attacker, the scheme could wait a number of queries before banning the flagged user. For example, if the scheme were to wait until the attacker's number of queries hits the next power of 2 to ban their account, and no user makes more than 2^{20} queries, then a cancellation can reveal only up to $lg(21) = 4.4$ bits of information about the detector and its threshold. This mechanism would also increase the cost of the extracting the encoder.

Chapter 8

Zero Query Defense

Attack	Success Rate	Num. of Queries	Similarity Detections
NES (Best)	17%	22,510±17,000	442
Boundary (Best)	95%	19,928±24,300	391
Query Surrogate (Untargeted)	5%	8,192+/-0	82

Table 8.1: **Attack success and detection rate for EAT-defended model.** Our defense still detects query based attacks frequently, while the defended model is now robust to zero query attacks. Each attack could make up to 200,000 queries, and was run on the same set of randomly selected 100 images and target classes. Each metric is the average over the over the most successful attacks.

We now explore complementing our query-based scheme with previous works in defenses against zero query attacks. Currently, one of the most effective defenses against black box, zero query attacks is ensemble adversarial training (EAT) [36], in which a defended model is retrained on white-box adversarial examples, with maximum distortion ϵ generated on an ensemble of static models with different architectures and weights. This training procedure has been demonstrated to then make the defended model robust against adversarial examples transferred from a holdout model, while resulting in only a reasonable decrease in the defended model's clean accuracy (on non-adversarial examples). Accordingly, we train a ResNet50v1 classifier for CIFAR-10 using EAT, with $\epsilon = 0.05$, to construct a model robust to zero query attacks of $\epsilon = 0.05$ (further training details can be found in the appendix.)

To evaluate the EAT-defended model, ResNet50v1-EAT, against zero query transfer attacks, we generated adversarial examples (over the CIFAR-10 test set) on a holdout ResNet74v2 model, and then saw if those examples transferred (i.e. were also adversarial to the ResNet50v1-EAT). The ResNet74v2 model was trained on the same CIFAR-10 training set, with the same training parameters, which gives the attacker their best case of information to train a surrogate. To generate the examples, we used FGSM[17] and the a clipped modification of the ℓ_2 version of the Carlini-Wagner (CW) attack [9] (with $\epsilon = 0.05$

Model	Clean	FGSM	CW	CW (Targeted)
ResNet50v1	7.8	73.8	59.1	18.8
ResNet50v1 (EAT)	14.6	14.4	16.4	1.0

Table 8.2: Error rate and attack success rate on an unprotected ResNet compared to an EAT defended ResNet [36].

and $\kappa = 100$ for the Carlini-Wagner attack). The error rate of the defended ResNet50v1-EAT are shown below, and all attacks are untargeted unless specified otherwise.

Compared to the undefended model, the EAT-defended model is noticeably more robust to all three methods of transfer attack. The EAT-defended model does incur a noticeable decrease in clean (test set) accuracy, but this comes as the tradeoff for the robustness to a fairly large value of $\epsilon = 0.05$ (for reference, [36] used $\epsilon = 0.06$ for defending models trained for much higher dimensional, 256x256x3 Imagenet images). In practice, the defender may tune ϵ according to their demands between accuracy and robustness.

We now show that our defense is still successful for an EAT-defended model. We reran the best query based attack variants on this EAT-defended model, and results of these attacks are shown in Table 8.1. Our scheme is still able to detect the query based attacks frequently, and it appears that the EAT model may even mitigate the success rate of query based attacks.

Chapter 9

Related Work

To our knowledge, our scheme is the first to use query distances in order to detect query based black box attacks. [23] explores using a distance based detection scheme to detect model extraction attacks (attempts to construct a surrogate version of a target model by actively querying the target model). Their scheme flags detection based on the adherence of the distribution of distances between queries to a normal distribution; however, their scheme is not robust to natural circumventions such as inserting dummy queries, or even applying our query blinding attack.

Previous work has focused on stateless detection of a given query as adversarial, usually by various means of determining if the query is out of the distribution of normal/benign data [29, 14, 19]. However, consistent and effective detection under this stateless threat model has proven difficult [8].

Other work has been done to defend against white-box attacks, such as adversarial training [28]. Such defenses are complementary to our defense: we can apply our detection strategy on top of any model. In our paper we study the defense on top of a non-robust model for simplicity and to accurately measure the value of this type of defense.

There are other query based black box attacks, but they often follow either a similar gradient estimation approach to NES, or a similar boundary following approach to the Boundary Attack. For example, SPSA [38] is another common gradient estimation attack, that estimates the gradient over random instead of Gaussian directions. Boundary attack variants, like Boundary++ [22], RED [24], and qFool [27], are more query efficient but still follow the same core approach of querying along the boundary.

Transfer attacks are a common approach in the zero query setting [31]. We explore the application of ensemble adversarial training [36], currently one of the most effective defenses against zero-query transfer attacks, to our scheme, but recent key-based defense may also be effective [33].

The approach for query blinding takes inspiration from previous work in signature blinding [10] and mimicry attacks [40].

Chapter 10

Limitations and Future Work

We see vast possibilities for future work in this research direction.

Our defense by design explicitly assumes that the model weights are able to be held secret. If the model weights are revealed to the attacker (e.g., through a model extraction attack [37]) then our proposed defense again is not effective. There has been related work [23] which explicitly defends against this type of attack. These types of defenses are complementary to ours and both could be applied simultaneously.

Our proposed defense only prevents attacks which attempt to generate a *specific* adversarial example. If the adversary were content on finding *any* error, then an adversary could simply generate random candidate inputs and wait until a test error randomly occurred [15]. For example, because our CIFAR-10 classifier reaches 92% accuracy an adversary who wanted to identify *any* error would be expected to succeed after showing just 13 images. This attack is not specific to our defense, and would work equally well on any other defense to adversarial examples.

The specific auto-encoder attack transformation we develop is explicitly designed to target the defense we have constructed, and likely future defenses could detect this specific attack strategy. However, we believe the general query-blinding strategy is an interesting research direction to pursue to develop stronger attacks.

Our proposed defense was only evaluated in the "hard-label" setting where the adversary receives only the classification label, without confidence scores. Designing a defense in the "soft-label" black-box setting, where the model does return confidence scores, is an interesting direction for future work. In Appendix E we give a proposed attack which which demonstrates the potential difficulties in solving this case.

It may be possible to train a stronger defense, which achieves higher robustness against our autoencoder-based query-blinding attack, by performing *adversarial training*. Similar to Generative Adversarial Networks [16], we have a similarity encoder neural network which is designed to detect similar images, and a autoencoder that is designed to produce images that fool the similarity detector. It is possible that performing adversarial training may lead to a robust similarity encoder.

Chapter 11

Conclusion

Black-box defenses to adversarial examples are most widely studied because of their realistic threat model with practical real-world implications. In such situations, the academic community has thus far artificially restricted research in this space to stateless defenses; we argue that expanding the study to stateful defenses gives the defender a new potential advantage. Towards this end, we propose a simple query-detector which detects the process of adversarial example generation. By combining our proposed approach with existing defenses which prevent transferability attacks, we can construct the first unified defense that can argue robustness in the full black-box threat model.

Support

This work was supported by generous gifts from Google and Futurewei and by the Hewlett Foundation through the Center for Long-term Cybersecurity.

Chapter 12

Appendix

12.1 Similarity Encoder

Layer	Filters	Size	Details
Conv	32	3×3	ReLU
Conv	32	3×3	ReLU
Max Pool		2×2	stride = 2
Dropout			$p = 0.25$
Conv	64	3×3	ReLU
Conv	64	3×3	ReLU
Max Pool		2×2	stride = 2
Dropout			$p = 0.25$
Dense		512	ReLU
Dropout			$p = 0.5$
Dense		256	

Table 12.1: **Encoder Architecture**

12.2 Transformations

Parameters

For the low distortion transformations (expected $\ell_2 = 2.32$), their parameters (r) are the following:

For the selected high distortion transformations (expected $\ell_2 = 5.33$) their parameters (r) are the following:

Transform	r
Uniform Noise	0.064
Translate	0.45
Rotate	0.018
Pixel-wise Scale	0.17
Crop and Resize	0.04
Brightness	0.09
Contrast	0.55
Gaussian Noise	0.095

Table 12.2: **Transformation parameters (low distortion)**

Transform	r
Pixel-wise Scale	0.36
Brightness	0.204
Contrast	0.79

Table 12.3: **Transformation parameters (high distortion)**

Further Experiments

We also explored reducing s, the number of samples per confidence score estimation, to $s = 2$ versus the original value $s = 50$, as a lower s should result in less groups of $k = 50$ neighboring points being significantly near each other and accordingly less detections. This comes at a cost of potentially less accurate estimations of the confidence scores, so the adjustment of s is experimented with in combination with modifying the confidence score estimation procedure. Additionally, a higher value of s increases the chance that the attack will succeed, at the cost of increasing the number of detections due to the corresponding higher number of queries. The comparison of $s = 2$ versus $s = 50$ over different image transformations is shown in Figure 12.1.

12.3 Hybrid Query Based Surrogate

For this attack, we selected D as $D = 0.09$, by experimentally finding the necessary value such that samples would on average have a higher similarity encoding distance than the threshold, and $d = 0.01$. We used the same three layer CNN described in Table 12.1 as our architecture for the surrogate model, and trained it for 50 epochs per attack.

$s = 2$:	Success Rate	Num. of Queries	ℓ_2 Detections	Sim. Detections
Uniform Noise	1%	15,695	308	308
Translate	4%	6,714	131	131
Rotate	7%	10,175	199	198
Scale	27%	12,567	246	246
Crop and Resize	7%	6,132	119	119
Brightness	55%	13,503	263	264
Contrast	23%	11,197	219	219
Gaussian Noise	0%	n/a	n/a	n/a
$s = 50$:	Success Rate	Num. of Queries	ℓ_2 Detections	Sim. Detections
Uniform Noise	15%	453,651	8,895	8,895
Translate	28%	293,586	5,755	5,756
Rotate	50%	295,022	5,782	5,784
Scale	83%	337,346	6,607	6,614
Crop and Resize	43%	249,717	4,895	4,895
Brightness	45%	277,864	5,437	5,447
Contrast	23%	253,325	4,966	4,966
Gaussian Noise	0%	n/a	n/a	n/a

Figure 12.1: **Attack success and detection rate of an ℓ_2 and similarity encoding based distance metric.** For each image transformation, the NES attack was run 100 times using that transformation to generate samples for estimating confidence scores, with $s = 2$ or $s = 50$ samples drawn per estimate and allowing up to $200,000$ or $5,000,000$ queries respectively. Each attack was run on the same set of randomly selected 100 images and target classes, and each metric is the average over the successful attacks.

12.4 Ensemble Adversarial Training

For training a CIFAR-10 classifier with EAT, we start with a ResNet50v1 (pretrained to a CIFAR-10 accuracy of 92.2%) and then train it on adversarial examples generated on an ensemble of trained ResNets with different architectures: ResNet44v1, ResNet56v2, and ResNet74v1. Adversarial examples were generated using the Fast Gradient Sign Method (FGSM) [17], with $\epsilon = 0.05$ (same as the query based attacks). The network was trained for 100 epochs, where the adversarial examples for each epoch were generated from a randomly selected model from the ensemble and the defended model, and one adversarial example was generated per image in the CIFAR-10 training set.

12.5 A Difficult Soft-Label Case

Our paper focuses on the "hard-label" threat model where the adversary only is only given the `arg mac` prediction. In contrast, the "soft label" threat model gives the adversary full access to the output probabilities of the model.

Similar to hard-label black-box attacks, soft-label black-box attacks rely on querying the value $f(x)$ for various x in order to generate their attack. Applying the above types of transformations would be possible (however, this time optimizing to make the difference between the neural network outputs small, $f(x) - f(\alpha(x))$).

However, below we present one possible attack that demonstrates the increased capabilities for an attacker when given access to the probability distribution. In particular, this attack shows that our hard-label defense scheme does not scale to the soft-label setting.

To obtain the classification of a sample x we define the functions $f(\cdot)$ and $r(\cdot)$ as follows:

- Select a random unit-vector direction r; for a distance hyperparameter d, let

$$f(x) = \{x + r \cdot d, x + r \cdot (d+]\epsilon)\}$$

- Recover the output by letting

$$r(y_0, y_1) = y_0 + (y_0 - y_1) \cdot \frac{d}{\epsilon}$$

This attack works due to the fact that neural networks, despite being non-linear functions globally, often locally behave as if they were linear functions.

To increase the accuracy of the estimate of $f(x)$ the above procedure can be generalized by selecting multiple random directions r_i and averaging the results. We experimentally verified that this type of attack approach is effective and allows the confidence-variant of NES [21] to succeed against our query-based detector.

Bibliography

[1] Wieland Brendel *, Jonas Rauber *, and Matthias Bethge. Decision-based adversarial attacks: Reliable attacks against black-box machine learning models. In *International Conference on Learning Representations*, 2018.

[2] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[3] Anish Athalye and Nicholas Carlini. On the robustness of the CVPR 2018 white-box adversarial example defenses. *CoRR*, abs/1804.03286, 2018.

[4] Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018*, July 2018.

[5] Sean Bell and Kavita Bala. Learning visual similarity for product design with convolutional neural networks. *ACM Trans. on Graphics (SIGGRAPH)*, 34(4), 2015.

[6] 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.

[7] Thomas Brunner, Frederik Diehl, Michael Truong-Le, and Alois Knoll. Guessing smart: Biased sampling for efficient black-box adversarial attacks. *CoRR*, abs/1812.09803, 2018.

[8] Nicholas Carlini and David Wagner. Adversarial examples are not easily detected: Bypassing ten detection methods. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pages 3–14. ACM, 2017.

[9] Nicholas Carlini and David A. Wagner. Towards evaluating the robustness of neural networks. *CoRR*, abs/1608.04644, 2016.

[10] David Chaum. Blind signatures for untraceable payments. In *Advances in cryptology*, pages 199–203. Springer, 1983.

[11] François Chollet et al. Keras. https://keras.io, 2015.

[12] Clarifai. Transforming enterprises with computer vision ai (image recognition service). *https://clarifai.com/*.

[13] Logan Engstrom, Andrew Ilyas, and Anish Athalye. Evaluating and understanding the robustness of adversarial logit pairing. November 2018.

[14] Reuben Feinman, Ryan R. Curtin, Saurabh Shintre, and Andrew B. Gardner. Detecting adversarial samples from artifacts. *CoRR*, abs/1703.00410, 2017.

[15] Nic Ford, Justin Gilmer, Nicolas Carlini, and Dogus Cubuk. Adversarial examples are a natural consequence of test error in noise. *arXiv preprint arXiv:1901.10513*, 2019.

[16] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc., 2014.

[17] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*, 2015.

[18] Google. Google cloud vision ai (image recognition service). *https://cloud.google.com/vision/*.

[19] Kathrin Grosse, Praveen Manoharan, Nicolas Papernot, Michael Backes, and Patrick D. McDaniel. On the (statistical) detection of adversarial examples. *CoRR*, abs/1702.06280, 2017.

[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[21] Andrew Ilyas, Logan Engstrom, Anish Athalye, and Jessy Lin. Black-box adversarial attacks with limited queries and information. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018*, July 2018.

[22] Michael I. Jordan Jianbo Chen. Boundary attack++: Query-efficient decision-based adversarial attack. *https://arxiv.org/abs/1904.02144*, 2019.

[23] Mika Juuti, Sebastian Szyller, Alexey Dmitrenko, Samuel Marchal, and N. Asokan. PRADA: protecting against DNN model stealing attacks. *CoRR*, abs/1805.02628, 2018.

[24] Faiq Khalid, Hassan Ali, Muhammad Abdullah Hanif, Semeen Rehman, Rehan Ahmed, and Muhammad Shafique. Red-attack: Resource efficient decision based attack for machine learning. *CoRR*, abs/1901.10258, 2019.

[25] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[26] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research).

[27] Yujia Liu, Seyed-Mohsen Moosavi-Dezfooli, and Pascal Frossard. A geometry-inspired decision-based attack. *CoRR*, abs/1903.10826, 2019.

[28] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[29] Jan Hendrik Metzen, Tim Genewein, Volker Fischer, and Bastian Bischoff. On detecting adversarial perturbations. In *Proceedings of 5th International Conference on Learning Representations (ICLR)*, 2017.

[30] 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.

[31] 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.

[32] Joseph Redmon and Ali Farhadi. YOLO9000: better, faster, stronger. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 6517–6525, 2017.

[33] Ilia Shumailov, Xitong Gao, Yiren Zhao, Robert Mullins, Ross Anderson, and Cheng-Zhong Xu. Sitatapatra: Blocking the transfer of adversarial samples. *CoRR*, abs/1901.08121, 2019.

[34] David Silver, Aja Huang, Christopher J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel,

and Demis Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529:484–503, 2016.

[35] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations*, 2014.

[36] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. Ensemble adversarial training: Attacks and defenses. In *International Conference on Learning Representations (ICLR)*, 2018.

[37] Florian Tramèr, Fan Zhang, Ari Juels, Michael K. Reiter, and Thomas Ristenpart. Stealing machine learning models via prediction apis. In *25th USENIX Security Symposium (USENIX Security 16)*, pages 601–618, Austin, TX, 2016. USENIX Association.

[38] Jonathan Uesato, Brendan O'Donoghue, Aäron van den Oord, and Pushmeet Kohli. Adversarial risk and the dangers of evaluating against weak attacks. *CoRR*, abs/1802.05666, 2018.

[39] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. 2017.

[40] David Wagner and Paolo Soto. Mimicry attacks on host-based intrusion detection systems. In *Proceedings of the 9th ACM Conference on Computer and Communications Security*, pages 255–264. ACM, 2002.

[41] Weilin Xu, David Evans, and Yanjun Qi. Feature squeezing: Detecting adversarial examples in deep neural networks. *CoRR*, abs/1704.01155, 2017.

[42] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *CoRR*, abs/1605.07146, 2016.