Stateful Detection of Black-Box Adversarial Attacks

Steven Chen
University of California, Berkeley

Nicholas Carlini
Google Research

David Wagner
University of California, Berkeley

ABSTRACT

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

This paper argues that in such settings, defenders have a larger space of actions than previously studied. 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 evaluate the space of stateful defenses.

We develop a defense designed to detect the process of *generating* adversarial examples. 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.

CCS CONCEPTS

• **Security and privacy** → *Intrusion detection systems*; • **Computing methodologies** → *Neural networks*; *Computer vision.*

KEYWORDS

detection, neural networks, adversarial examples, black box attack

ACM Reference Format:
Steven Chen, Nicholas Carlini, and David Wagner. 2020. Stateful Detection of Black-Box Adversarial Attacks. In *Proceedings of the 1st Security and Privacy on Artificial Intelligent Workshop (SPAI '20), October 6, 2020, Taipei, Taiwan.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3385003.3410925

1 INTRODUCTION

Defending neural networks against adversarial examples has proven to be extremely difficult. Most published defenses have been found to have significant flaws [2, 9], and even the few defenses that have withstood validation offer only partial robustness [30]. Adversarial examples can even be generated in a fully *black-box* threat model, in which an adversary can only query the model and receive the predicted classification as output. While there are domains where the adversary has white-box access to a deployed neural network, in many production environments where neural networks are deployed, the user can only query the classifier and observe the output. For example, services such as Clarifai [14] and Google Cloud Vision AI [21] offer image classification APIs where users can submit images and receive only the label of that image.

This paper studies 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 [9]. We consider the task of identifying the *sequence* of queries made to the classifier when creating an adversarial example. Based on the observation that existing black-box attacks make a sequence of highly self-similar queries (i.e., each query in the sequence is similar to prior queries in the sequence), we develop a defense that uses a similarity-detector neural network to identify such query patterns, and find that the existing state-of-the-art black-box attack algorithms can be detected through this strategy. Our defense composes with existing defenses for defense-in-depth.

Then, we develop the first adaptive attacks against stateful defenses and evaluate the robustness of the proposed defense in the practical use case of image classification APIs. We propose *query blinding*, a general strategy for attacking defenses which monitors 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 blinded inputs match benign data patterns, but (2) it is possible to deduce the classifier's output from these queries. We validate the efficiacy of query blinding by showing it breaks PRADA (Euro S&P'19) [25]—a stateful defense to model stealing attacks. We show that our stateful 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.

We make the following contributions:

- We propose a new class of adversarial example defenses: stateful detection defenses.
- We design and evaluate a defense in this category, and find it is effective at detecting existing attacks and is hard to evade even when the attacker adapts the attack approach.
- We introduce *query blinding*, a general strategy that can be used to attack stateful detection defenses.
- We release the source-code for our defense and attacks at **https://github.com/schoyc/blackbox-detection**.

2 BACKGROUND & PROBLEM STATEMENT

This paper studies evasion attacks on neural networks [4, 35]. Following most prior work on the space of adversarial examples [2], we evaluate on image classifiers. Images are represented as $h \cdot w \cdot c$ dimensional vectors (with height h, width w, and c color channels). In this paper we use ℓ_∞ distance as the metric $\|\cdot\|$ for measuring

SPAI '20, October 6, 2020, Taipei, Taiwan

ACM ISBN 978-1-4503-7611-2/20/10.
https://doi.org/10.1145/3385003.3410925

adversarial distortions; this is the most common metric used in the space of images [19]. Here, $\ell_\infty(x, x') = \max_i |x_i - x'_i|$, the maximum difference between any component. We expect our results will naturally extend to other ℓ_p-based metrics.

Threat model. We defend against *hard-label* black-box attacks, where an adversary can only query a model; the parameters are unavailable. For example, a machine-learning-as-a-service provider might allow users to query a model after creating an account, but not allow downloading the model. 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 might be able to create as many accounts as they wish, but there is a cost associated with creating each account (e.g., linking to a valid credit card and email address, paying an account fee, etc.).

In our scheme, the attacker's account is 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 through the number of times the attack is detected before it is successful (number of accounts that must be created); this determines the attacker's cost to defeat the system. (e.g. on the black market, a stolen credit card with CVV number costs $5 [7, 16], so a single attack requiring 100 accounts would cost at least $500.)

The *hard-label* setting means that model queries return only the categorical labels assigned by the classifier, but not the numerical confidence scores associated with it. Our approach extends naturally to other settings, but as argued in prior work [4] we believe the hard-label setting is the most realistic black-box threat model.

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.

Query Attacks. Our defense is motivated by the sequential nature of hard-label query-based black box adversarial attacks, such as NES [24] and the Boundary Attack [5]. 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 points near the current proposed adversarial example.

Considering attack queries as a sequence, successive queries are likely to be close together (by some distance metric), because (1) each iteration of the attack makes a small gradient-estimation step or boundary-following 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

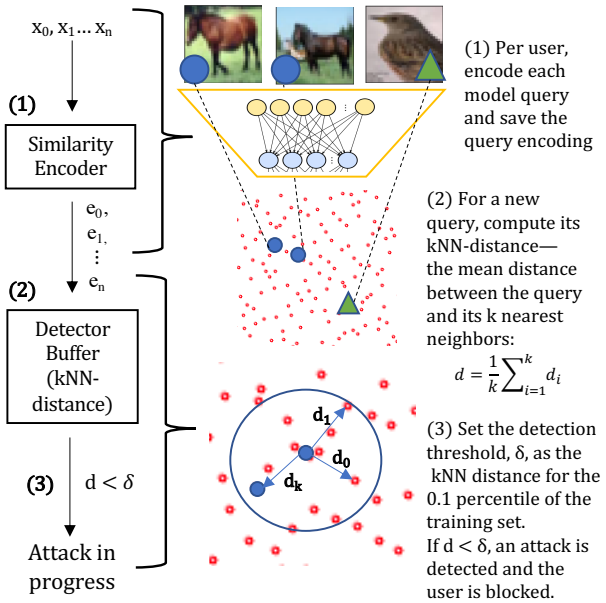

Figure 1: Our defense that detects query-based attacks.

the sequence of queries made to a model can detect an attack based on an anomalous pattern of suspiciously close queries.

Zero-query attacks. 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* [19] to the other. This observation motivated the earliest black-box attack algorithms: train a "surrogate model" [32] on the same task as the target model, perform a gradient-based 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.

By definition, without query sequences to monitor, our approach cannot defend against transfer attacks and other zero-query attacks. Fortunately, 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 zero-query defenses (e.g., EAT) is that they are not effective against query-based attacks. Thus, this 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.

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

At a high level, our defense is applied as an access monitor on top of an existing classifier. The 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 (tuned according to the defender's resources, see Section 7). 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).

We measure the distance between queries by encoding the queries using a similarity encoder [3], to map perceptually similar images to nearby points in a reduced dimensional space, and then apply ℓ_2 distance. If the mean distance falls below a threshold, we flag this query as an attempt at generating an adversarial example. The detection threshold is set so that under benign use (i.e. if the entire training set were to be randomly streamed as queries) 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 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) Simple metrics, such as ℓ_2, do 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 the ℓ_2 norm, which experimentally we find allows an adversary to evade detection.
(2) The ℓ_2 distance requires storing an entire copy of every query. This could pose a significant cost to the hosting service in storage costs, and storing user queries for longer than strictly necessary introduces potential privacy risks.

To increase the security of our scheme, we use perceptual similarity for nearest-neighbor search, as adversarial attacks involve generating new images that are 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.

[1] In practice, a lower false positive rate may be necessary. However, existing defense research for detecting adversarial examples sets the false positive rate at approximately 5% [39] (NDSS'18). Our value is thus 50× lower than prior work.

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 $x \in \mathbb{R}^{h \cdot w \cdot c}$ to an encoded space $e \in \mathbb{R}^d$ of dimension d. As described, the objective of this encoder is to map visually similar inputs $x, \tilde{x}$ to encodings $e = E(x)$ and $\tilde{e} = E(\tilde{x})$ that are similar under ℓ_2 distance, so that $\|e - \tilde{e}\|_2$ is small.

To achieve this we train the similarity encoder neural network with a contrastive loss function [3]. Specifically, we consider two pairs of images. Pair 1 consists of x_i, an image drawn from the training set, and x_p, a "positive" image perceptually similar to x_i. Pair 2 consists of a different training image x_j, along with a negative example x_n, an image *not* perceptually similar to x_j. The contrastive loss for their encodings $(e_i, e_p), (e_j, e_n)$ is

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

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.

Following prior work [3], we then use this loss function to fine-tune a classification network (trained on the same set of images), modified to replace the final logits layer with a d-dimensional encoding layer. Positive training pairs were generated by applying a random image transformation (that should retain the image's perceptual content) to batches of training set images, while negative pairs were simply two randomly selected training images. The transformations used are enumerated in Section 5.1, and further training details can be found in the appendix.

3.3 Experimental Setup & Parameter Selection

We evaluate our defense on the CIFAR-10 dataset as is done in most other adversarial example work [1, 30].

The choice of k, the number of neighbors, affects the effectiveness of our scheme. Large values might improve the effectiveness of our scheme at detecting attacks (since larger k allows a larger threshold while maintaining the 0.1% false positive rate, thus forcing the attacker's images to be very different to avoid detection). However, k is the minimum number of queries before our defense could possibly flag a possible attack, so smaller values of k will enable faster detection of attacks. Accordingly, we plotted the threshold for a 0.1% FPR as a function of k, and found that this threshold increases sharply until $k = 50$, where the distance begins to plateau and marginally continues to increase as k increases. Thus, we use $k = 50$ for evaluating our scheme.

With $k = 50$ and the corresponding threshold $\delta = 1.44$, we evaluate the FPR of our defense against the CIFAR-10 test set and find the FPR to be 0%. We also evaluate against CINIC-10 [15] (210,000 images of the CIFAR-10 classes taken from the ImageNet dataset and downsampled to $32 \times 32 \times 3$), and find the FPR to be 0.3%. This is slightly higher than the targeted FPR of 0.1%, likely due to the slightly different distribution of the CINIC-10 images, but still very reasonably close to the desired FPR.

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-based attacks. While there are many black-box (hard-label) attacks, they fall into two categories:

- **Gradient estimation** attacks operate like standard white-box gradient-based attacks. However, because they do not have access to the gradient, these types of attacks instead estimate the gradient by repeatedly querying the model.
- **Boundary following** attacks, in contrast, first identify the decision boundary of the neural network (possibly far away), and then take steps following the boundary to locate the nearest point on the boundary to the target image.

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, where the original image is a test set image that is correctly classified by the targeted model. We call an attack successful if the ℓ_∞ distortion is below $\varepsilon = 0.05$. Most white-box work on CIFAR-10 considers the smaller distortion bound of $\varepsilon = 0.031$ [1]. We choose a larger distortion bound because black-box attacks are known to be more difficult to generate and so we give the adversary more power to compensate. The network we attack is a ResNet [23] trained on the CIFAR-10 dataset for 100 epochs with Adam [28] and achieves 92% test accuracy.

NES [24] is one of the two most prominent gradient-estimation attacks (along with SPSA [37]). It estimates the gradient by averaging the confidence scores of randomly sampled nearby points, and then uses projected gradient descent [30] to perturb an image of 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 a class's score as the fraction of those points of that class.

The Boundary Attack [5] 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 [6, 12]. We evaluate our defense on the vanilla boundary attack; the other attacks are more query efficient, but at their core 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) as implemented in FoolBox [33].

4.2 Results

We run each attack against our scheme and find that they can be detected. The results are presented in Table 1. An attack is considered successful if an adversarial example is found within an ℓ_∞ distortion of $\epsilon = 0.05$ from the original image, with the correct target class. We terminate each attack as soon as it finds such an example. Each attack does eventually succeed at a high rate, but is detected frequently: 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 detect un-modified attacks.

Table 1: Success rate of unmodified attacks on a neural network protected with our scheme. While these attacks are successful, 100% of attacks are detected, with each attack instance requiring hundreds or thousands of accounts on average (over 100 instances per attack).

Attack	Detected	Attacked	Queries	Accounts
NES	100%	100%	325,200±153,300	6,377
Boundary	100%	100%	14,720±8,923	288

5 QUERY BLINDING: AN ADAPTIVE ATTACK

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 [8].

Thus, we introduce *query blinding*, a general strategy which can be used to hide the query sequence from the defender. Query blinding is the most effective attack strategy we have found against our scheme. At its core, the objective of a query blinding attack is to learn the value of $f(x)$, for some specific x, without actually revealing the example x to the defender. 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* $r(f(x'_0), f(x'_1), \ldots, f(x'_n))$ that is designed to estimate $f(x)$ from the classifier outputs. For the majority of this paper we restrict ourselves to the case where $\|b(x)\| = 1$.

5.1 Image Transformations for Query Blinding

Let x be the image that an attacker would like to query the model for, $f(x)$ be the model's output, and $T_c(x; r)$ be a randomized image processing transform (e.g., by rotating or shifting the image by a random amount c); then we set $b(x; r) = \{T_c(x; r)\}$. In order to successfully fool the detector, we would like the distortion between the original image and the transformed image to be large. For example, adjusting the brightness of a CIFAR-10 image by adding 0.05 to each pixel introduces an ℓ_2 distortion of $0.05 \times \sqrt{3 \times 32 \times 32} = 2.77$. After distortion, these transformations still retain the primary content of the image, and a model with high accuracy should produce relatively similar outputs for the original and transformed images, so the corresponding revealing function for an image processing transformation is simply $r(f(x')) = f(x')$. We examined seven possible transformations: adding uniform noise, image translation, image rotation, pixel-wise scaling, crop-and-resize, brightness adjustment, and contrast adjustment.

5.2 Auto-Encoder for Query Blinding

We also consider attacks that involve *learning* the blinding function. Specifically, we train an auto-encoder neural network $\alpha(x)$. Normally, auto-encoders are trained so that $\alpha(x) \approx x$. In our case we instead 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

Table 2: Query blinding breaks PRADA, but not our defense (averaged over 100 instances). An adversary needs 950 accounts to generate one adversarial example on our scheme.

Defense	Detected	Attacked	Queries	Accounts
PRADA	0%	47%	48,400±22,800	0
Ours	100%	47%	48,400±22,800	950

detection by the encoder, while property (2) ensures that the actual classification of the image will remain unchanged.

Specifically, we train the auto-encoder to minimize the loss

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

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

We train the auto-encoder 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 Evaluation

We now evaluate query blinding against PRADA [25], an existing detection scheme for model extraction attacks. (As far as we are aware, there are no other current schemes designed to detect black box adversarial attacks, so we found this to be the most relevant existing defense that could also potentially detect query based black box attacks.) Briefly, the crux of PRADA's scheme is to compute how closely the pairwise ℓ_2 distance between queries conforms to a normal distribution, and then compare that statistic to a threshold.

According to the process described by its authors, we tune the parameters of PRADA such that it also achieves a 0.1% FPR on the CIFAR10 training set (see Appendix C.1). We use the pixel-wise scale transformation (scaling all pixels by $c \sim U(1-r, 1+r); r = 0.34$) for query blinding with the NES attack ($s = 5$ and $\sigma = 0.1$), and find that the attack goes undetected, as seen in Table 2. This shows that even the simpler query blinding of using image transformations is effective at bolstering attacks and evading existing defenses. This modified NES attack is explained further in-depth in Section 6.1.

6 ADAPTIVE ATTACK EVALUATION

Given that our proposed defense effectively prevents existing black-box attacks, we now study whether or not it can prevent more sophisticated attacks. We find that while it is possible to degrade the effectiveness of the defense, we can not defeat it completely. We study both gradient attacks (specifically, variants of NES with various kinds of query blinding) and boundary-following attacks (specifically, variants of the boundary attack with query blinding).

6.1 The NES Attack

To generate a targeted adversarial example for a given input x, NES starts with an image x' of the target class, t, and then uses projected gradient descent to reduce the distortion between this image (already of the target class) and the original example x to be within ϵ of the original image. Per iteration of projected gradient descent, the NES attack makes use of two procedures that require queries (full details can be found in [24]):

(1) **Gradient approximation**: Sample n basis points, θ_i, within some ℓ_∞ distance σ of x; then average their n confidence scores to approximate the gradient, $\nabla P(y = t|x)$.
(2) **Confidence score estimation**: For each θ_i, sample s queries within an ℓ_∞ radius μ of θ_i, and estimate θ_i's score for class t as the proportion of the s queries that were of class t.

6.1.1 Parameter Tweaking. The default attack parameters for NES are $\sigma = 0.001$, $n = 4$, $s = 50$, $\mu = 0.001$, and learning rate = 0.01 [24]. We try to adjust these parameters to make it harder for our defense to detect the attack. We increase μ, the radius of the sampling ball used when sampling points for confidence score estimation, and find the attack succeeds up to $\mu = 0.064$. At this value, the 50-nearest-neighbor distance between an image and the s sampled points is on average 2.32, significantly larger than the 50-nearest-neighbor distance of just 0.032 for $\mu = 0.001$.

We then decrease s and generate fewer queries near θ_i when estimating confidence scores. The attack remains reasonably successful even reduced to $s = 2$, while the attack becomes harder to detect, so we use $s = 2$ here on. The attacker could also increase σ so that the sampled Gaussian basis points θ_i are further apart, and we leverage this in the high-distortion attacks in the next section.

6.1.2 Query Blinding. We modify the confidence score estimation procedure to sample s points using the transformations listed in Section 5.1 instead of sampling from a ℓ_∞ ball of uniform radius. The parameters for each transformation are normalized so that the expected ℓ_2 distortion from each transformation is equal to 2.32. [2] When running the NES attack, each time we query the classifier we preprocess the image with one strategy. Table 3 shows the effectiveness of different transformations. For some transformations, like uniform and Gaussian noise, the NES attack fails completely. However, the NES attack works even better with brightness and pixel-scale transformations than the original confidence estimation procedure of uniform noise. This suggests that estimating the confidence score for an image may be more accurate with certain image transformations than others.

For all transformations, each attack requires at least one hundred accounts (on average), so our defense is effective at detecting these query blinding attacks. The exact attacker cost corresponding to this number of accounts is quantified further in Section 7.

For this level of transformation distortion, the similarity encoder offers little benefit over ℓ_2 distance. This is understandable, as when $k = 50$ and using ℓ_2 distance as the distance metric for the defense, the ℓ_2 detection threshold is $\delta = 5.069$, which is greater than the ℓ_2 distortion of 2.32 induced by these transformations. We also

[2] We selected a constant of 2.32 to match the ℓ_2 distortion of setting $\mu = 0.064$. (Note that μ is now only applicable when using the original strategy of sampling from a ball of ℓ_∞ radius μ). We used the same parameters when training the similarity encoder.

Table 3: Success rate of query blinding using NES and Boundary Attack. The number of accounts an adversary would need to generate a single adversarial example varies between 97 and 504 for NES, and 76 to 609 with the Boundary/HopSkipJump Attack. The ℓ_2 detector performs strictly worse than our simiarlity encoder, especially on auto-encoder attacks. (Each attack was run over the same 100 randomly selected images and target classes, with a budget of 200,000 queries per attack instance.)

	Attack	Attacks Detected	Attack Success Rate	Queries	Similarity Detector Accounts	ℓ_2 Detector Accounts
Low Distortion	NES (query blinding: uniform noise)	100%	1%	15,700±0	308	308
	NES (query blinding: translate)	100%	4%	6,710±400	131	131
	NES (query blinding: rotate)	100%	7%	10,200±684	198	199
	NES (query blinding: scale)	100%	27%	12,600±882	246	246
	NES (query blinding: crop and resize)	100%	7%	6,130±247	119	119
	NES (query blinding: brightness)	100%	55%	13,500±780	264	263
	NES (query blinding: contrast)	100%	23%	11,200±777	219	219
High	NES (query blinding: brightness)	100%	43%	24,500±2,630	481	60
	NES (query blinding: scale)	100%	42%	25,700±3,000	504	88
	NES (query blinding: contrast)	100%	37%	19,100±4,240	375	85
	NES (query blinding: auto-encoder)	100%	76%	13,400±8,400	97	12
	Boundary (normal)	100%	100%	14,700±892	288	-
	Boundary (Perlin noise)	100%	40%	31,100±976	609	-
	Boundary (query blinding: brightness)	100%	2%	24,100±1,470	473	-
	Boundary (query blinding: auto-encoder)	100%	61%	34,200 ± 8,000	240	-
	HopSkipJumpAttack	100%	100%	3,940 ± 3,480	76	-

evaluated against attacks with transformations that introduce a greater distortion. We increase σ to $\sigma = 0.1$ (the highest value found experimentally where the attack still succeeds reasonably often) and increase the distortion parameter for each transformation so that the expected ℓ_2 distortion introduced is 5.10. Results for these higher-distortion attacks are shown in Table 3.

Against these higher-distortion transformations, the similarity encoder is 6–8× more effective at detecting attacks compared to ℓ_2 distance on images. As explained in Section 7, this significantly increases the cost to the attacker. This demonstrates that the similarity encoder significantly increases robustness against adaptive query blinding attacks. The number of accounts needed increases when using higher-distortion transformations versus lower-distortion transformations; this is because the attack needs more queries on average to succeed (due to less precise gradient estimates), resulting in more opportunities to detect the attack.

The learned auto-encoder is more effective than simpler transforms. (Visualizations of auto-encoded images are presented in Figure 2, in the Appendix.) The auto-encoder is particularly effective at defeating a defense that uses ℓ_2 distance on images, as it is trained to map an image to a blinded version with large ℓ_2 distortion. However, our full defense with a similarity encoder remains able to detect the auto-encoder attack. Future work may be able to improve the auto-encoder attack by training it adversarially.

6.2 The Boundary Attack

The boundary attack [5] 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 two operations. 1) *Distortion-reducing (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 $x'_{i+1} = x'_i + \varepsilon \cdot (x - x'_i)$. 2) *Orthogonal (boundary-following) step*, starting from the current adversarial example x'_i, take a step along a random direction r orthogonal to $x - x'_i$, subject to the constraint that $f(x'_i + r) = f(x'_i)$.

We evaluate our defense against different variations of the boundary attack. First, we sample random directions from Perlin noise instead of a Gaussian distribution, as this substitution was shown to increase the attack's success rate with a limited number of queries (fewer than 15,000 queries) [6]. We also evaluate the effectiveness of a boundary attack with query blinding. In particular, we preprocess all queries made by the boundary attack with the transformation that worked best with the NES attack (brightness), as well as the autoencoder. Lastly, we evaluate against the HopSkipJumpAttack (HSJA) [12], an improved extension of the boundary attack that uses significantly fewer queries by estimating the direction of the gradient with binary information at the boundary. We use the default parameters provided in the authors' public implementation.

Table 3 shows our ability to detect different versions of the boundary attack. We allow 200,000 queries, the same number of queries as the best NES attack variants. Perlin noise does not perform better than the original boundary attack, likely due to the observed decreasing utility of the Perlin noise at these higher query numbers [6]. Preprocessing queries with the brightness transform is also ineffective: it does not decrease the number of accounts needed while significantly decreasing the boundary attack's success rate.

This is because the boundary attack adjusts its step size according to the local geometry of the boundary (estimated from past queries); the preprocessing causes the attack to misapproximate the local boundary geometry and converge prematurely before finding a point within distance ϵ from the original image x and failing to make further progress. Consequently, of the variants we studied, only HSJA is more effective than the original boundary attack.

Our defense is effective at detecting the boundary attack: an attacker would need to create 200 accounts to create a single adversarial example. Even using the HopSkipJumpAttack, an attacker would require at least 75 accounts to generate a single successful adversarial example when our detector is in place.

6.3 Multi-Account Sybil Attack

So far our attacker has only used one account at a time in order to generate an adversarial example. However, an attacker might try to create multiple accounts and then distribute their queries among these multiple accounts. In Appendix C.3, we evaluate the efficacy of our defense at detecting an adversary who creates multiple Sybil accounts and uses them to construct one attack. In our naive construction, *cyclic account reuse*, an attacker creates C accounts and then to generate an adversarial example, queries the i-th query through account $i \mod C$. However, we find that this attack (as well as a more powerful one) does not succeed.

6.4 Attacking the Similarity Encoder

An attacker might try to defeat our scheme by fooling the similarity encoder, e.g., generating a blinded version x' of a query x such that the similarity encoder considers x, x' to be dissimilar, yet they have the same classification. Because the similarity encoder is kept secret in our defense, there is no way to perform a white-box attack to construct such blinded queries. There is no direct way to mount a black-box query attack, either: there is no way to supply an image x and learn its encoding, or submit a pair of images x_0, x_1 and learn whether they have similar encodings.

This leaves only side-channel attacks on the similarity encoder. For instance, an attacker could create a new account, submit a batch of $k+1$ images, and observe whether the account is cancelled; this reveals whether the $k+1$st image was similar to the previous k. However, such an attack would be very slow. Each $k+1$ queries reveal only a single bit of information about the similarity encoder. As constructing a surrogate model or mounting query-based attacks typically require tens or hundreds of thousands of examples, such an attack would require creating an infeasible number of fake accounts.

Information-theoretically, to maximize the amount of information revealed per query, the optimal strategy is to issue $k = 50$ queries, then issue subsequent queries chosen so that each has about a $p \approx 1/18$ chance of triggering detection and observe which one causes the account to be cancelled; this reveals about 5.6 bits of information per account and issues about 68 queries per account (on average). We expect that it would still require thousands of fake accounts, so it is unlikely to be effective.

7 ECONOMICS OF PERFORMING AN ATTACK

Assuming our defense was in place, what would the economics look like for an attacker who wished to completely avoid detection, using a single account? We consider a thought experiment where an attacker is given information for whether or not each query to the detection scheme will be detected as part of an attack sequence and knows k. Normally, the attacker will not know this information, as the encoder is not made public in our scheme.

Case 1: time-bounded buffer. Consider an attack that needs d accounts, i.e., is detected d times before successfully generating an adversarial example. If we store each query until t hours have passed, then an attacker with one account would need dt hours to execute the attack without detection. The most effective attack against our scheme that we have found (HopSkipJumpAttack) triggers 76 detections. If we store queries for 100 hours, generating a single adversarial example without being detected would take over 10 months.

An attacker may also make benign queries constantly in order to simply impose a high resource cost to the defender to maintain the time bounded buffer. However, in practice, most machine learning APIs/services already enforce a rate limit on users that not only prevents the original service from abuse/overuse but also in this case upper bounds the resource cost of the buffer.

For example, Google's Cloud Vision API currently has a rate limit of 1800 queries per minute per user, so a user who continuously makes the maximum number of queries could make 10.8M queries per 100 hours. Our similarity encoder maps each input to a 256-dimensional output, so storing these 10.8M vectors (with 16-bit floating point values) would require 5.5GB of storage. Google's cloud storage is currently priced at $0.026/GB/month per user [20], so the storage cost to the provider of the query buffer would be at most $0.14/month per user—and only if those users spent over $2,000 per month on API requests. Thus, even for a reasonably long buffer duration of 100 hours, the defender incurs a relatively small monetary cost in the worst case of a maximally querying user, and an even smaller cost in the average case.

Case 2: query-bounded buffer. Suppose we always store the last N queries made by each user. An attacker who knows which queries will trigger a detection could avoid detection by flushing the buffer (by making N extra random queries) just before being detected. An attack that triggers d detections (i.e., normally requires d accounts) could be executed by an attacker with a single account in this setting using about dN total queries if the attacker wishes to avoid being detected. Making these additional "hiding" queries to disguise the attack comes at a significant cost to the attacker.

For example, Google's Cloud Vision API currently costs $1.50 USD per 1000 queries [20]. If the buffer stores $N = 10^4$ examples, our most effective attack would require about 76×10^4 queries to execute without being detected, which would cost about $1150 USD, a sizeable amount of money for an adversary to pay for generating a single adversarial image. For comparison, without our defense, the best attack we tried would cost about $6 USD. At current Google cloud storage prices, the cost to the service provider to running the defense, for $N = 10^4$, would be about $0.0007/month per user.

8 ZERO QUERY DEFENSE

Our scheme detects query-based attacks, but cannot detect zero-query attacks. We propose that our scheme be combined with ensemble adversarial training (EAT), one of the most effective existing

Table 4: Effectiveness of our defense with an EAT-defended model. Against the most effective attack variants (NES with low distortion brightness query blinding, the Boundary attack with normal sampling, and the HopSkipJumpAttack), our defense still detects query-based attacks, while EAT makes the defended model robust to zero-query attacks. The attacks' success rates also decrease for the EAT-defended model compared with the non EAT-defended model. 100 instances were run per attack, and each attack instance could make up to 200,000 queries. Queries and accounts needed are an average over the successful attack instances.

Attack	Attack Success Rate (EAT)	(non-EAT)	Queries	Accounts
NES (best)	17%	55%	22,510±17,000	442
Boundary	95%	100%	19,928±24,300	391
HSJA	96%	100%	7,390±4,970	143

Table 5: Robustness against transfer attacks of an unprotected ResNet compared to an EAT-defended ResNet. The second column shows the error rate on clean examples; the subsequent columns show the success rate of transfer attacks using untargeted FGSM, and untargeted and targeted variants of the attack from Carlini and Wagner [10].

Model	Clean	FGSM(u)	CW(u)	CW(t)
ResNet50v1	7.8%	73.8%	59.1%	18.8%
ResNet50v1-EAT	14.6%	14.4%	16.4%	1.0%

defenses against black-box zero-query attacks [36]. EAT generates white-box adversarial examples with distortion ϵ on an ensemble of static models with different architectures and weights, and trains the defended model on these examples. This procedure has been demonstrated to make the defended model robust against adversarial examples transferred from a holdout surrogate model, while resulting in only a modest 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 for the ResNet50v1-EAT). To give the attacker every advantage, we trained the ResNet74v2 surrogate model on the same CIFAR-10 training set, with the same training parameters. We used FGSM [19] and a clipped modification of the Carlini-Wagner (CW) ℓ_2 attack [10] to generate adversarial examples (with $\epsilon = 0.05$ and $\kappa = 100$ for the CW attack). Table 5 shows the success rate of attacks against the defended ResNet50v1-EAT.

Compared to the undefended model, the EAT-defended model is more robust to all three transfer attacks. The EAT-defended model does incur a noticeable decrease in clean accuracy, but in exchange we obtain substantial robustness against transfer attacks for 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.

Our defense remains effective an EAT-defended model. We reran the best query-based attack variants on this EAT-defended model; results are shown in Table 4. Our scheme is still able to detect query-based attacks frequently, and it appears that the EAT model may even reduce the success rate of query-based attacks.

9 LIMITATIONS

The main limitation we see for our defense is with video classification, as benign, sequential video frames are likely to be very similar to each other. To evaluate this use case, we sampled frame sequences from three videos that focused on CIFAR10 classes (e.g. footage of the National Dog Show) to input into our scheme. Since the threshold, δ, was originally set according to the non-self-similar CIFAR10 training set, we tune δ to tolerate the increased similarity for benign images. All attacks were detected from k-neighbors distances well below the original threshold set, so we lower the threshold to the lowest value such that all attacks are still detected.

With this adjustment, we found the FPR on these videos could range up to 3.8%, higher than our targeted FPR of 0.1%, but still lower than prior work [39], even in one of the most challenging settings for this defense. However, in settings where users classify video frames frequently, a nonzero FPR may not be practical and further tuning or development may be necessary (see Appendix E).

10 CONCLUSION

Defenses against white-box adversarial examples have thus far proven elusive; thus, we advocate for increased study into black-box defenses against adversarial examples. In the black-box setting, the academic community has thus far studied only stateless defenses; we argue that stateful defenses give the defender a new advantage and deserve attention. Towards this end, we propose a simple scheme that detects the process of adversarial example generation, and introduce query blinding, an adaptive attack on such schemes that is effective at breaking prior work. By combining our proposed approach with existing defenses that prevent transferability attacks, we construct the first unified defense that might offer black-box robustness.

ACKNOWLEDGMENTS

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

REFERENCES

[1] Anish Athalye and Nicholas Carlini. 2018. On the Robustness of the CVPR 2018 White-Box Adversarial Example Defenses. *CoRR* abs/1804.03286 (2018). arXiv:1804.03286 http://arxiv.org/abs/1804.03286

[2] Anish Athalye, Nicholas Carlini, and David Wagner. 2018. Obfuscated Gradients Give a False Sense of Security: Circumventing Defenses to Adversarial Examples. In *ICML*. https://arxiv.org/abs/1802.00420

[3] Sean Bell and Kavita Bala. 2015. Learning Visual Similarity for Product Design with Convolutional Neural Networks. *ACM Trans. on Graphics (SIGGRAPH)* 34, 4 (2015).

[4] Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Srndic, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. 2017. Evasion Attacks against Machine Learning at Test Time. *CoRR* abs/1708.06131 (2017). http://arxiv.org/abs/1708.06131

[5] Wieland Brendel, Jonas Rauber, and Matthias Bethge. 2018. Decision-Based Adversarial Attacks: Reliable Attacks Against Black-Box Machine Learning Models. In *ICLR*. https://openreview.net/forum?id=SyZI0GWCZ

[6] Thomas Brunner, Frederik Diehl, Michael Truong-Le, and Alois Knoll. 2018. Guessing Smart: Biased Sampling for Efficient Black-Box Adversarial Attacks. *CoRR* abs/1812.09803 (2018). arXiv:1812.09803 http://arxiv.org/abs/1812.09803

[7] BuyAccs.com. [n.d.]. Black market site to buy email accounts in bulk. https://buyaccs.com/en/buy-bulk-gmail-accounts.php?partner=pvabuy.

[8] Nicholas Carlini, Anish Athalye, Nicolas Papernot, Wieland Brendel, Jonas Rauber, Dimitris Tsipras, Ian Goodfellow, and Aleksander Madry. 2019. On evaluating adversarial robustness. *arXiv preprint arXiv:1902.06705* (2019).

[9] Nicholas Carlini and David Wagner. 2017. Adversarial examples are not easily detected: Bypassing ten detection methods. In *ACM Workshop on Artificial Intelligence and Security*. 3–14.

[10] Nicholas Carlini and David A. Wagner. 2016. Towards Evaluating the Robustness of Neural Networks. *CoRR* abs/1608.04644 (2016). arXiv:1608.04644 http://arxiv.org/abs/1608.04644

[11] David Chaum. 1983. Blind signatures for untraceable payments. In *CRYPTO*. Springer.

[12] Jianbo Chen, Michael I. Jordan, and Martin J. Wainwright. 2019. HopSkipJumpAttack: A Query-Efficient Decision-Based Adversarial Attack. *arXiv:1904.02144* (2019).

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

[14] Clarifai. [n.d.]. Computer Vision and AI Enterprise Platform. https://clarifai.com/.

[15] Luke Nicholas Darlow, Elliot J. Crowley, Antreas Antoniou, and Amos J. Storkey. 2018. CINIC-10 is not ImageNet or CIFAR-10. (2018). arxiv.org/abs/1810.03505

[16] Experian. [n.d.]. Here's How Much Your Personal Information Is Selling for on the Dark Web. https://www.experian.com/blogs/ask-experian/heres-how-much-your-personal-information-is-selling-for-on-the-dark-web/.

[17] Reuben Feinman, Ryan R. Curtin, Saurabh Shintre, and Andrew B. Gardner. 2017. Detecting Adversarial Samples from Artifacts. *CoRR* abs/1703.00410 (2017). arXiv:1703.00410 http://arxiv.org/abs/1703.00410

[18] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative Adversarial Nets. In *NIPS*. Curran Associates, Inc. http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf

[19] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and Harnessing Adversarial Examples. In *ICLR*. http://arxiv.org/abs/1412.6572

[20] Google. [n.d.]. Cloud Storage Pricing. https://cloud.google.com/storage/pricing.

[21] Google. [n.d.]. Google Cloud Vision AI. https://cloud.google.com/vision/.

[22] Kathrin Grosse, Praveen Manoharan, Nicolas Papernot, Michael Backes, and Patrick D. McDaniel. 2017. On the (Statistical) Detection of Adversarial Examples. *CoRR* abs/1702.06280 (2017). arXiv:1702.06280 http://arxiv.org/abs/1702.06280

[23] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep Residual Learning for Image Recognition. *CoRR* abs/1512.03385 (2015). arXiv:1512.03385 http://arxiv.org/abs/1512.03385

[24] Andrew Ilyas, Logan Engstrom, Anish Athalye, and Jessy Lin. 2018. Black-box Adversarial Attacks with Limited Queries and Information. In *ICML*. https://arxiv.org/abs/1804.08598

[25] M. Juuti, S. Szyller, S. Marchal, and N. Asokan. 2019. PRADA: Protecting Against DNN Model Stealing Attacks. In *2019 IEEE European Symposium on Security and Privacy (EuroS P)*. 512–527.

[26] keras.io. [n.d.]. github.com/keras-team/keras/blob/master/examples/cifar10_cnn.py.

[27] Markus Kettunen, Erik Härkönen, and Jaakko Lehtinen. 2019. E-LPIPS: Robust Perceptual Image Similarity via Random Transformation Ensembles. arxiv:1906.03973.

[28] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *ICLR*. http://arxiv.org/abs/1412.6980

[29] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. [n.d.]. CIFAR-10. http://www.cs.toronto.edu/~kriz/cifar.html

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

[31] Jan Hendrik Metzen, Tim Genewein, Volker Fischer, and Bastian Bischoff. 2017. On Detecting Adversarial Perturbations. In *ICLR*. arxiv.org/abs/1702.04267

[32] Nicolas Papernot, Patrick D. McDaniel, Ian J. Goodfellow, Somesh Jha, Z. Berkay Celik, and Ananthram Swami. 2016. Practical Black-Box Attacks against Deep Learning Systems using Adversarial Examples. *CoRR* abs/1602.02697 (2016). arXiv:1602.02697 http://arxiv.org/abs/1602.02697

[33] Jonas Rauber, Wieland Brendel, and Matthias Bethge. 2017. Foolbox: A Python toolbox to benchmark the robustness of machine learning models. In *Reliable Machine Learning in the Wild Workshop, 34th International Conference on Machine Learning*. http://arxiv.org/abs/1707.04131

[34] Ilia Shumailov, Xitong Gao, Yiren Zhao, Robert Mullins, Ross Anderson, and Cheng-Zhong Xu. 2019. Sitatapatra: Blocking the Transfer of Adversarial Samples. *CoRR* abs/1901.08121 (2019). arXiv:1901.08121 http://arxiv.org/abs/1901.08121

[35] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2014. Intriguing properties of neural networks. In *ICLR*. http://arxiv.org/abs/1312.6199

[36] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. 2018. Ensemble Adversarial Training: Attacks and Defenses. arXiv preprint arXiv:1705.07204. In *ICLR*. https://arxiv.org/abs/1705.07204

[37] Jonathan Uesato, Brendan O'Donoghue, Aäron van den Oord, and Pushmeet Kohli. 2018. Adversarial Risk and the Dangers of Evaluating Against Weak Attacks. *CoRR* abs/1802.05666 (2018). arXiv:1802.05666 http://arxiv.org/abs/1802.05666

[38] David Wagner and Paolo Soto. 2002. Mimicry attacks on host-based intrusion detection systems. In *CCS*. ACM.

[39] Weilin Xu, David Evans, and Yanjun Qi. 2017. Feature Squeezing: Detecting Adversarial Examples in Deep Neural Networks. *CoRR* abs/1704.01155 (2017). arXiv:1704.01155 http://arxiv.org/abs/1704.01155

A ADDITIONAL RELATED WORK

As described in Section 5.3, the most closely related work is to ours is PRADA [25]. However, this work does not perform adaptive attacks on their scheme, and we show that query blinding is able to defeat this defense. Moreover, while our defense is robust against Sybil attacks, their authors note that PRADA is not. They do not consider how to detect creation of adversarial examples.

Much previous work has explored stateless detection of whether an individual query is adversarial, usually by checking if the query is out of the distribution of normal/benign data [17, 22, 31]. However, effective detection under this stateless threat model has proven difficult [9]. We instead examine whether access to prior state allows the defender to gain an advantage.

Developing robust classifiers (absent the black-box threat model) is a large research direction. One of the best known approaches is adversarial training [30]. Any of these such defenses are complementary to our detection scheme: we can apply our detection strategy on top of any model. In our paper we study our defense on top of a non-robust model for simplicity and to accurately measure the value of this type of defense. Recent work on robust similarity [27] could also be useful for improving our defense.

There are other query-based black-box attacks that may be useful for performing better query blinding attacks. These attacks often follow either a similar gradient-estimation approach as NES, or a boundary-following approach similar to the boundary attack. For example, SPSA [37], another gradient-estimation attack, estimates the gradient with Bernoulli instead of Gaussian directions.

Transfer attacks are a common approach in the zero-query setting [32]. We explore combining our defense with ensemble adversarial training [36], currently one of the most effective defenses against zero-query transfer attacks, but the recent Sitatapatra defense may also be effective [34].

The approach for query blinding takes inspiration from previous work in signature blinding [11] and mimicry attacks [38].

B SIMILARITY ENCODER TRAINING

As in [3], we initialize our encoder with the same architecture and weights as a network trained to classify the desired images. For CIFAR-10, we train a three-layer CNN [26] for 100 epochs using data augmentation and reach a validation accuracy of 76%. We then substitute the logits layer with a new encoding layer of

dimension $d = 256$. We found a margin of $m = \sqrt{10}$ experimentally resulted in the best encodings. A more complex architecture might yield stronger results, but to keep our initial design simple we use this CNN. We fine-tune this modified network to minimize the contrastive loss function described in Section 3.2, with a learning rate $\alpha = 1e^{-4}$, momentum $\mu = 0.9$, and batch size $b = 32$.

C ADAPTIVE ATTACKS/EVALUATION

C.1 PRADA Evaluation Parameters

PRADA sets a single parameter, the detection threshold, δ, according to a desired false positive rate (FPR). As described in [25], PRADA's FPR is evaluated by dividing a set of benign queries into chunks of 50 queries, inputting each chunk into PRADA, and computing the FPR as the fraction of chunks that trigger detection. Similar to tuning our scheme's threshold, we use the CIFAR10 training data as the set of benign queries, randomly divide it into 1,000 chunks, and tune the threshold to $\delta = 0.88$ to yield an FPR of 0.1%.

C.2 Additional Figures

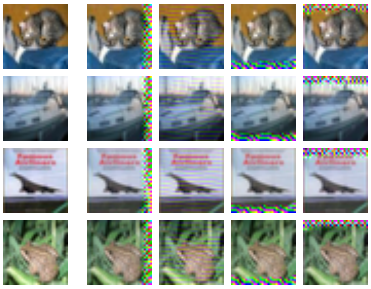

Figure 2: For each image in the test set (left) we show four possible images transformed by our auto-encoder (right) that have the same classification label but have high ℓ_2 distance from each other.

C.3 Extended Sybil Account Attack

Cyclic account reuse strategy. In the simplest Sybil attack strategy, an attacker creates C accounts and then, for a given attack instance, makes the i-th query through account $i \mod C$. The primary benefit of such a strategy is that any given detector is only allowed to observe every C-th query. An attacker may hope that this will increase the distance between consecutive queries seen by the detector (per account) and could require fewer accounts.

However, we find that this does not actually help. Upon investigation, we find the reason for this is that in practice, the randomness introduced through gradient estimation is roughly at the same scale as the *total* perturbation introduced throughout the process, and therefore observing sequential queries is much less important than observing queries from the same adversarial example generation process. However, this is just one attack strategy: a careful attacker may come up with a stronger one.

Near-optimal account reuse strategy. We now show that it is unlikely there exists a Sybil strategy which partitions a query sequence to reduce the detection rate by more than a factor of two.

We perform the following setup. We take the query sequence $\{x_i\}_{i=1}^{N}$ for a run of NES (using brightness for query blinding) to generate an adversarial example x_N and process each input with the encoder to obtain the embeddings $e_i = e(x_i)$. Then, we compute the pairwise distance between all pairs of points $d_{i,j} = \|e_i - e_j\|$.

We now ask: what is the largest set $S \subset [1..N]$ such that the k-nearest neighbor distance for each point is larger than the detection threshold. Because finding the largest set is NP Hard (it is easy to see through reduction from maximum clique), we approximate this quantity greedily, starting with the empty set and adding elements that are maximally far apart. On performing this experiment, we find that when a defense uses k-nearest neighbor distance, it is not possible to construct a set with $|S| > 2 \cdot k$. (For example, when $k = 5$ the largest set we can construct is $|S| = 9$; when $k = 25$ the largest set we can construct is $|S| = 44$.) Thus, if we conservatively assume that every set could be made this largest size, we are guaranteed one detection at least every $2k$ queries per Sybil account, reducing the detection rate by at most a factor of two. We now discuss further disadvantages of using Sybil accounts.

Drawback of Sybil Accounts. For anti-abuse reasons, most services already take measures to ensure that users do not create a large number of accounts. However, further, if a single user created multiple accounts and spread queries across each of these accounts, we believe this would only make it easier for the service provider to detect that one user was using multiple Sybil accounts. For example, if occasionally the service provider ever performed an across-user query-history analysis, it would be possible to discover the same user making highly-similar queries across different accounts.

D ENSEMBLE ADVERSARIAL TRAINING

We pre-train a ResNet50v1 on CIFAR-10 (accuracy 92.2%), then train it for 100 epochs on adversarial examples generated on an ensemble of different trained ResNets: ResNet44v1, ResNet56v2, and ResNet74v1. Examples were generated using FGSM [19] with $\epsilon = 0.05$, and each epoch's adversarial examples were generated from a randomly selected model from the ensemble and the defended model (one example generated per CIFAR-10 training image).

E VIDEO CLASSIFICATION

Videos were sampled at 30 frames per second, then downsized to $32x32x3$, yielding 7,000 frames per video. The threshold was tuned from $\delta = 1.44$ to $\delta = 0.7$. Decreasing the sampling rate by a factor of C decreased the FPR by a factor greater than C, (e.g. reducing the frame rate from 30fps to 15fps reduced the FPR by 0.6 instead of 0.5), as successive frames were less likely to have similar content.

The defender may need to further tune the threshold according to the natural workload that their system would expect (e.g. if the classification system is to be used for video frames, then a set of video frames, rather than static images, should be used to set the defense's threshold). The video classification setting may also be significantly expensive for users, as such frequent querying to an image classification API would incur high costs.