Blacklight: Scalable Defense for Neural Networks against Query-Based Black-Box Attacks

Huiying Li
University of Chicago

Shawn Shan
University of Chicago

Emily Wenger
University of Chicago

Jiayun Zhang*
Fudan University

Haitao Zheng
University of Chicago

Ben Y. Zhao
University of Chicago

Abstract

Deep learning systems are known to be vulnerable to adversarial examples. In particular, query-based black-box attacks do not require knowledge of the deep learning model, but can compute adversarial examples over the network by submitting queries and inspecting returns. Recent work largely improves the efficiency of those attacks, demonstrating their practicality on today's ML-as-a-service platforms.

We propose *Blacklight*, a new defense against query-based black-box adversarial attacks. Blacklight is driven by a fundamental insight: to compute adversarial examples, these attacks perform iterative optimization over the network, producing queries highly similar in the input space. Thus Blacklight detects query-based black-box attacks by detecting highly similar queries, using an efficient similarity engine operating on probabilistic content fingerprints. We evaluate Blacklight against eight state-of-the-art attacks, across a variety of models and image classification tasks. Blacklight identifies them all, often after only a handful of queries. By rejecting all detected queries, Blacklight prevents any attack from completing, even when persistent attackers continue to submit queries after banned accounts or rejected queries. Blacklight is also robust against several powerful countermeasures, including an optimal black-box attack that approximates white-box attacks in efficiency. Finally, we illustrate how Blacklight generalizes to other domains like text classification.

1 Introduction

The vulnerability of deep neural networks (DNNs) to a variety of adversarial examples is well documented. An adversarial example is a maliciously modified input that looks (nearly) identical to its original via human perception, but gets misclassified by a DNN model. This vulnerability remains a critical hurdle to the practical deployment of deep learning systems in safety- and mission-critical applications, such as autonomous driving or financial services.

*Work done while visiting SAND Lab at the University of Chicago.

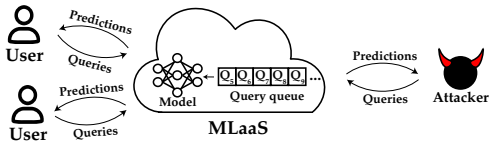

Figure 1: *Attack Scenario for black-box adversarial attacks.*

Adversarial attacks can be broadly divided by whether they assume *white-box* or *black-box* threat models. In the *white-box* setting, the attacker has total access to the target model, including its internal architecture, weights and parameters. Given a benign input, the attacker can directly compute adversarial examples as an optimization problem. In contrast, an attacker in the *black-box* setting can only interact with the model by submitting queries and inspecting returns. Black-box scenarios can be further divided based on the information the classifier returns per query: *score-based* systems return a full probability distribution across labels, and *decision-based* systems return only the output label.

The white-box threat model makes a strong assumption: an attacker has obtained total access to the model, through a server breach, a malicious insider, or other type of model leak. Both security and ML communities have made continual advances in both attacks and defenses under this setting – powerful attacks efficiently generate adversarial examples [12, 16, 27, 37, 71], which in turn spur work on robust defenses that either prevent the generation of adversarial examples or detect them at inference time. While numerous approaches have been explored as defenses (e.g., model distillation [55], gradient obfuscation [8, 21, 47, 61, 65, 77], adversarial training [49, 82, 83], honeypots [62], and ensemble methods [68]), nearly all have been proven vulnerable to followup attacks [4, 9–11, 28].

In contrast, black-box attacks assume a more realistic threat model, where attackers interact with models via a query interface such as ML-as-a-service platforms [81] (See Fig 1). There are two types of black-box attacks. Most common are *query-based attacks* [5, 14, 15, 30, 53, 69], where an attacker iteratively adapts the query input based on past query

results from the target model, until it produces a successful adversarial example. Numerous efforts have developed increasingly efficient attacks that require fewer queries to complete the attack. Unfortunately, even as these attacks grow in efficiency and practicality, there exists no effective defense against them. Existing defense proposals [17, 34] focus on detecting (and banning) query accounts displaying some "adversarial" behaviors. While raising the attack cost, they are ineffective against persistent attackers who switch accounts to evade detection and complete the attack. The second type of black-box attacks is *substitute model attacks*, where an attacker queries the target model to train a local model, then tries to transfer adversarial examples from the substitute to the target [45, 56, 57]. These are currently addressed by a line of effective and evolving defenses, including (ensemble) adversarial training [68, 75].

In this work, we focus on defending against query-based black-box attacks, even when persistent attackers switch account to evade detection. The fundamental insight driving our work is that, in order to compute adversarial examples, query-based black-box attacks *perform iterative optimization over the network*, an incremental process that produces queries highly similar in the input space. With this in mind, we propose *Blacklight*, a novel defense that detects query-based black-box attacks using an efficient *content-similarity engine*. Blacklight detects the highly similar queries as part of the iterative optimization process in the attack[1], since benign queries rarely share this level of similarity. Blacklight's query detection is account oblivious, thus is effective no matter how many accounts an attacker uses to submit queries.

Blacklight is highly scalable and lightweight. It detects highly similar queries generated by iterative optimization using *probabilistic fingerprints*, a compact hash representation computed for each input query. We design these fingerprints such that queries highly similar in the input space will have large overlap in their fingerprints. As such, Blacklight identifies an (incoming) query as part of a query-based black-box attack, if its fingerprint matches any prior fingerprint by more than a threshold. Since we use secure one-way hashes to compute fingerprints, even an attacker aware of our algorithm cannot optimize the content perturbation of a query to disrupt its fingerprint and avoid detection.

We evaluate the efficacy of Blacklight against eight SOTA query-based black-box attacks, including those using gradient estimation, gradient-free attacks, and those targeting score- and decision-based models. We experiment on a range of image-based models from MNIST to ImageNet, and use L_p distance metrics chosen by each attack. While these attacks typically take thousands (or tens of thousands) of queries to converge to a successful adversarial example, Blacklight detects all of them after the first 2–9 queries[2]. More importantly, Blacklight detects the large majority of all queries associated with an attack (e.g., >90% for all non-Boundary attacks). By rejecting these detected attack queries, Blacklight consistently reduces the attack success rate to 0% for all eight attacks, even when attackers persist to submit queries despite query rejection.

Our work makes the following key contributions.

- We propose a highly scalable, lightweight attack detection system against query-based black-box attacks, using probabilistic content fingerprint-based query matching to detect (and mitigate) individual attack query on the fly.
- We discuss and demonstrate why existing account-based defenses are insufficient to resist persistent attackers.
- We build formal analysis of our probabilistic fingerprints to model both attack detection rates and false positives.
- We experimentally evaluate Blacklight against eight SOTA black-box attacks on multiple datasets and image classification models. Not only does Blacklight detect all eight attacks, but it does so *quickly*, often after only a handful of queries, for attacks that would require several thousands of queries to succeed.
- We illustrate how Blacklight can be generalized beyond image classification, using text classification as an example.
- We finally evaluate Blacklight and show it is highly robust against a variety of adaptive countermeasures, including those allowing larger, human-visible perturbations. Blacklight performs well even against two types of *near-optimal* attacks: "query-efficient" attacks several orders of magnitude more efficient than current methods, and "perfect-gradient" attacks that approximate white-box attacks by perfectly estimating the loss surface at each query.

The source-code for Blacklight is available at `https://sandlab.cs.uchicago.edu/blacklight`.

2 Background on Black-box Attacks

As background, we briefly overview different types of black-box attacks and describe today's SOTA query-based black-box attacks (the focus of our work). We discuss existing defense proposals [17, 34] later in §4.

2.1 Overview of Black-box Attacks

Existing black-box attacks can be divided into two types: substitute model attacks and query-based black-box attacks. In this work, we target the latter.

Substitute Model Attacks. An attacker queries a target model repeatedly, uses the query results to build a labeled dataset and train a *substitute* model to approximate classification boundaries of the model. The attacker then generates

[1] In practice, even the most efficient black box attacks issue thousands of queries to generate a single attack, and nearly all such queries are constrained to be a small perturbation away from the benign input.

[2] The exception is the Boundary attack, which starts its query search with an image from the target label. Blacklight detects Boundary attacks after an average of less than 50 queries (see Table 2).

adversarial examples on the substitute model (using a white-box attack), hoping that they will succeed on the target model. This attack can successfully produce untargeted adversarial examples on small models like MNIST [56, 57], but is much less successful when producing targeted attacks or targeting larger models [45]. This spurs efforts to increase transferability between substitute and target models [22, 31, 44, 76, 78].

Defending against substitute model attacks is an active research area. Existing defenses include adversarial training [38], ensemble adversarial training [68], and adversarial training with single-step R+FGSM attack [75]. We note that ensemble adversarial training can be combined with Blacklight as a hybrid defense against both substitute model attacks and query-based attacks (details in the Appendix §A).

Query-Based Black-Box Attacks. A more common and effective attack is query-based black-box attacks. An attacker queries the target model repeatedly, often remotely over a network, to implement iterative optimization required to compute adversarial examples. Specifically, based on the past query results, the attacker iteratively perturbs the current query to produce the next query, hoping to converge to a successful adversarial example. Both gradient-estimation [14, 15, 18, 30, 69] and gradient-free algorithms [3, 5, 53] were developed to reduce the number of queries required to produce an adversarial example. While these attacks generally require thousands to hundreds of thousands of queries to produce a single adversarial example, they have proven to be effective, often achieving 100% success rate even against large models. In fact, recent efforts show that these attacks can already be successfully launched against real-world systems such as Google Cloud Vision API [30], Clarifai [5], and real applications like traffic sign and license plate recognition [25]. Finally, recent works also leverage substitute model-based priors when configuring queries [19, 29, 33, 67], which we also consider when evaluating Blacklight in §9.2.

2.2 SOTA Query-based Black-box Attacks

Our work targets query-based black-box attacks. We implement and test eight SOTA attacks (see Table 1). They cover both score- and decision-based attacks, and attacks relying on gradient estimation and those that do not. They all use L_p bounded perturbations, a prevailing attack setting.

	Gradient Estimation	**Gradient Estimation Free**
Score-based	NES - Query Limit [30]	ECO [53]
Decision-based	NES - Label Only [30] HSJA [14] QEBA [41] Policy-Driven [79]	Boundary [6] SurFree [51]

Table 1: *We consider eight query-based black-box attacks.*

NES (2 variants) [30]. NES enables efficient gradient estimation using far fewer queries and applies natural evolution strategies [73] to speed up the attack. NES has two variants: *NES query limit* for score-based models and *NES label-only* for decision-based models.

ECO [53]. Targeting score-based models, the attacker replaces gradient estimation with an efficient discrete surrogate, leading to faster convergence.

Boundary [6]. It is the first attack targeting decision-based models and does not use gradient estimation. To compute the adversarial example for x_0, the attacker starts from a random sample x from the target label t, iteratively adjusts x to "approach" x_0 while remaining being classified to t, until the difference between x_0 and x is within a predefined budget.

HSJA [14]. It augments Boundary [6] with gradient approximation. In each iteration, a 2-step gradient estimation is used to construct x_t that gets closer to the decision boundary, leading to much faster attack convergence than Boundary.

QEBA [41]. This is a variant of HSJA. Instead of estimating the full gradient vector, QEBA only estimates a core subset of the gradient vector.

Policy-Driven [79]). This is another recent attack built on top of HSJA. It applies a policy network to *learn* the best optimization direction at each step.

SurFree [51]. This gradient-free attack leverages certain geometrical properties to produce careful query trials along diverse directions near the decision boundaries.

3 Threat Model and Design Goals

In this work, we focus on defense against query-based black-box attacks for image classification. Our design principle should extend to other domains, which we demonstrate in §8.7 using text classification as an example. Here, we define our threat model, design goals and success metrics.

Attacker. The attacker queries a target model ($\mathbb{F}$) and uses the query results to craft adversarial examples against it, i.e., finding the perturbed version of a benign input x_0 that causes $\mathbb{F}$ to *misclassify* it to a target label t. To do so, the attacker repeatedly queries $\mathbb{F}$ with a sequence of n attack queries $x_1, \ldots, x_n$. The attack is successful if

$$\mathbb{F}(x_n) = t \quad \text{and} \quad ||x_n - x_0||_p < \varepsilon \tag{1}$$

where x_n is the computed adversarial example of x_0 and ε is the attacker's perturbation budget. Existing works show that a successful attack requires a large n, generally on the order of 10^3-10^6. Note that while focusing on prevailing attacks that bound perturbations by L_p distances, our defense should extend in principle to other query-based attacks (*e.g.,* patch, semantic attack). We discuss in §8.3 preliminary results on Sparse-RS [20], a query-based universal patch attack.

We make the following assumptions about the attacker:

- The attacker has no access to internal weights of $\mathbb{F}$ and can only send queries to obtain outputs of $\mathbb{F}$.
- The attacker has abundant computation power and resources to submit millions of queries.
- The attacker controls **multiple** accounts and IP addresses, and moves the attack across them if any IP addresses and/or

accounts are banned. Measurements have shown that attackers often utilize *Sybil* accounts [23, 40, 80].

- We begin with standard attackers who are unaware of Blacklight. Later in §9, we consider stronger adaptive attackers who apply countermeasures against Blacklight.

Defender. The defender hosts the target model $\mathbb{F}$. For each query, $\mathbb{F}$ can either return the full classification probability vector or only the classification label. We only make one assumption on the defender, that it has a **finite** amount of storage for use in attack detection. In practical terms, any defender storing state related to past queries has to periodically **reset** the storage, e.g., every 1 or 2 days, by clearing out the state of all past (benign) queries.

Design Goals. We target four key goals for our defense.

- The defense should detect attack queries with **high accuracy** and **high coverage**, while maintaining a **low false positive rate**. Since answering each attack query may leak model information, the defense should detect as many attack queries as possible.
- The defense should efficiently **scale** to industry production systems. For example, Facebook's content moderation systems process an average of 300M images per day, while those at Twitter process 340M tweets/day [13, 70].
- The defense should incur **low overhead** in terms of runtime (compared to model inference runtime) and storage.
- The defense must **resist persistent attackers** who can move between accounts, and/or continue submitting attack queries after account ban or query rejection.

4 Existing Defenses and Their Limitations

There are two known defenses against query-based black-box attacks: *Stateful Detection* (SD) [17] and PRADA [34]. Both are account-driven and focus on detecting/banning query accounts that submit attack queries. We now describe their detection methods, and discuss why these defenses (and their variations) are insufficient to resist persistent attackers covered by our threat model.

Stateful Detection (SD) [17]. SD inspects each query account to decide whether it is malicious or not. Given an account A and its queries submitted so far, SD examines whether these queries display "certain properties" related to computation of adversarial examples. Specifically, SD computes, for an incoming query q from A, the average pair-wise latent similarity between q and its k-nearest-neighbors in A's past queries. If the average latent similarity exceeds a threshold, SD flags A as adversarial. To compute the latent similarity, SD uses a pretrained similarity encoder to convert each query image into a latent space vector.

PRADA [34]. Originally designed to detect attacks that steal the target model, PRADA is shown to also detect query-based black-box attacks [17]. The key insight is that queries sent by an attacker are expected to have a characteristic distribution different from those of benign accounts. PRADA calculates the query distribution of each account based on the L_2 distance among queries, and defines a standard benign distribution computed from a set of benign queries. If an account A's query distribution shifts away from the standard benign distribution, PRADA labels A as malicious.

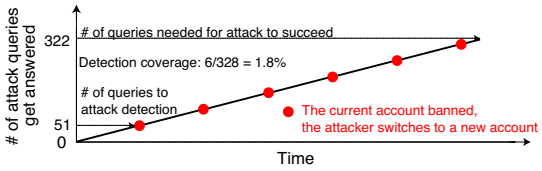

Figure 2: *Existing defenses cannot stop persistent attackers who switch accounts to continue attack queries.*

Vulnerability to Persistent Attacks. While SD and PRADA could flag an attacker who use a single account to send queries, they are ineffective against attackers holding multiple accounts, e.g. Sybil accounts [23]. Figure 2 plots an example where an attacker completes an attack, by switching accounts and continuing its queries after each detection by SD. A similar strategy would also succeed against PRADA.

The two existing defenses are limited by two factors. *First*, inspecting queries per-account puts a fundamental limit on detection speed, i.e., the number of attack queries answered by the model before detection. For both defenses, at the time of detection, the attacker already had tens or more attack queries answered by the model (e.g., 52 - 54 queries for SD and 111-115 queries for PRADA, per our experiments in Appendix Table 8). *Second*, both defenses are designed to "slow down" attackers by banning their current account rather than preventing the attack query to proceed. Given the low cost and prevalence of sybil accounts, attackers can easily bypass these defenses. A "reactive" strategy is shown in Figure 2, where 6 out of 328 attack queries (or 1.8%) were detected and rejected and 322 got answered. An alternative "proactive" strategy is to first run test cases to estimate the minimum # of attack queries to get the account banned (e.g., 50), and then during the attack, send less queries per account (e.g., 30) to evade detection completely.

Adapting Account-based Defenses. Account-based query inspection and mitigation is ineffective against attackers with multiple query accounts. An effective defense needs to be account oblivious. One straightforward solution is to run a version of SD or PRADA by putting all the queries into a single account. This solution, however, does not scale to support production ML systems facing millions of queries per day, because SD and PRADA's runtime complexity grows with the number of prior queries. Consider a query database of 1 million low-resolution images (`CIFAR10`), our experiments show that, for each incoming query, SD and PRADA introduce a run-time latency of 24,000% and 6,800% compared to the normal inference latency, respectively (details in §8.6). PRADA also faces large accuracy drop, because each incom-

ing query produces little impact on the query distribution.

5 Blacklight

We propose *Blacklight*, a new defense to detect and mitigate query-based black-box attacks against DNN models. Different from existing defenses, Blacklight is account oblivious and focuses on detecting individual attack queries on the fly regardless of who sent them. Our design is driven by a fundamental insight that query-based black-box attacks produce queries that are highly similar in the input space. Since benign queries rarely share this level of similarity, these attacks can be detected by identifying extremely high similarity in queries while incurring low false positives. With this in mind, we design Blacklight to focus on achieving fast, scalable and robust similarity check across millions of image queries. Our design includes two key components: (i) *probabilistic content fingerprinting* for fast and scalable attack detection, and (ii) *salted pixel quantization* to resist adaptive attacks.

In the following, we present the fundamental insight driving our design, and the concept of probabilistic content fingerprinting. Later in §6, we describe the salted pixel quantization and Blacklight's detailed design.

5.1 Fundamental Insight: Presence of High Similarity in Attack Queries

Blacklight exploits a fundamental insight on query-based black-box attacks: in order to compute adversarial examples, attackers need to perform iterative optimization *over the network*, i.e., submitting one or more queries to the target model, observing the query results, and using them to configure further queries. While the specific design of iterative optimization is algorithm-dependent[3], the unified goal is to repeatedly refine the perturbation such that the query sequence converges to an adversarial example x_n satisfying eq (1). Therefore, iterative optimization inevitably produces *some* queries that are highly similar in the input space, i.e.,

$$\text{there exist } x_k, x_j, \text{where } ||x_k - x_j||_p \leq \mu.$$

If μ is sufficiently smaller than the difference between most benign images, we can accurately detect the attack by recognizing the presence of highly similar queries like (x_k, x_j) within the stream of queries. Evading this type of detection is extremely difficult (if not infeasible) since it requires *every attack query to be sufficiently dissimilar from any previous attack queries*.

We empirically verify the presence of highly similar queries by running the eight SOTA query-based black-box attacks (listed in Table 1) on the `ImageNet` classification model. For all eight attacks, high similarity is consistently observed across images in their attack query sequence. The average L_2 distance between just consecutive queries in an attack sequence is already 20-380x smaller than analogous distance between benign images (estimated by randomly comparing 2000 pairs of benign images). Figure 3 shows some visual examples from attack query sequences generated by three attacks (NES-Query Limit, Boundary, HSJA). We omit the other attacks since they produce similar results.

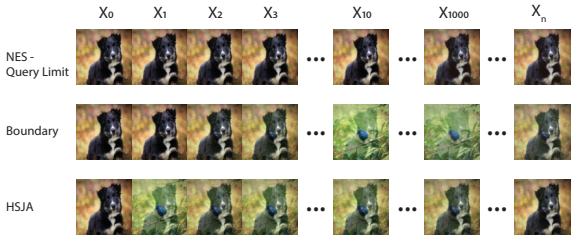

Figure 3: *Examples of attack query sequence* $(x_0, x_1, .., x_n)$*, produced by three black-box attacks (NES, Boundary, HSJA). While these attacks generate queries differently, the resulting query sequences all contain some highly similar images.*

5.2 Fast and Scalable Similarity Check via Probabilistic Fingerprinting

The above insight motivates us to detect query-based black-box attacks by searching for the presence of highly similar queries in a large stream of incoming and past queries. A key challenge is how to run a fast and efficient similarity check.

Strawman Solutions. We first discuss two strawman solutions and their problems. Earlier in §4 we discussed the query similarity check used by SD [17] and its scalability issue.

Computing L_p distances. A naive approach would store all past queries in a database and compare an incoming query x to the entire database of n queries by computing their image-level differences. Such raw comparison incurs heavy costs both in query storage and computation, i.e., $O(n)$. For example, even for low resolution image queries (224×224 pixels, `ImageNet`), it takes 23 minutes to compare a query to one million prior images, even using five threads on a 6-core Intel Xeon server. This is clearly intractable in practice.

Locality-sensitive (LS) hashing. An alternative is to compute a "signature" per query using LS hashes and compare queries by their signatures. Many have used perceptual hashing (*e.g.,* PhotoDNA [2], dhash [35]), a type of LS hashes, to match similar images for copyright resolution or child exploitation detection [2]. Using a hash table for lookup, the runtime cost for checking each incoming query could reach $O(1)$ regardless of n. Unfortunately, these hashes are designed to identify generic variants of an image, even those that have undergone significant alterations. Thus they flag similar benign queries (*e.g.,* different frames of a video, multiple pictures of the same object) as adversarial, producing false positives. We test dhash [35] on our attack detection and

[3] Some attack designs start with the original input and perturbs it towards a misclassified target label [30, 53], while others start from an image in the target label and work backwards towards the original input [6, 14, 30].

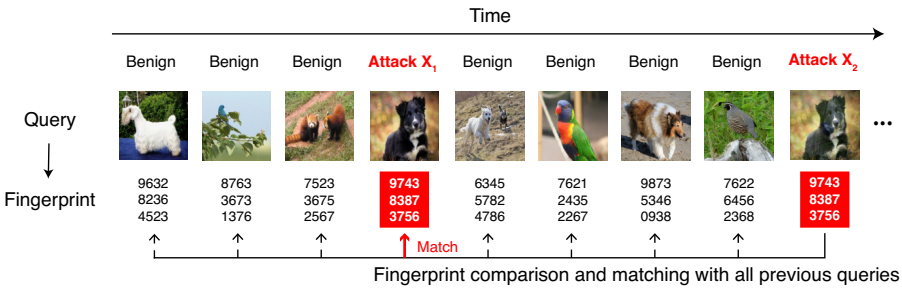

Figure 4: *For each raw image, Blacklight computes a small set of hash entries (as its probabilistic fingerprint). Blacklight detects attack images hidden inside a large stream of benign images by comparing and detecting highly similar fingerprints.*

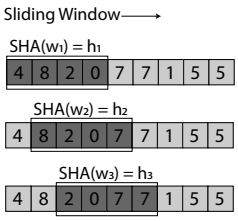

Figure 5: *Computing content hashes by applying a sliding window over pixels.*

find that it produces over 10% false positives on the Flickr dataset and 67% on our video dataset (§8.4). While unable to test PhotoDNA since it is proprietary, we expect that it faces the same issue since it focuses on detecting child exploitation in images which requires considerable alterations.

Probabilistic Fingerprints. Blacklight overcomes these challenges by applying probabilistic fingerprinting to detect highly similar images. Our goal is to design a hash function that is compact yet highly sensitive to very small changes in the image. This dictates that we should use a highly lossy function. Probabilistic fingerprinting achieves these properties and utilizes secure one-way hashes that cannot be easily reversed to evade detection and probabilistic downsampling for efficiency. To fingerprint an image x, Blacklight first transforms x into a set of continuous and overlapping segments of a fixed length $\mathbf{w}$, then applies a one-way hash to each segment to produce a large set of $\mathbf{N}$ hash values. From these $\mathbf{N}$ hash values, Blacklight chooses a small set probabilistically (e.g., the top 50) as x's probabilistic fingerprint.

Figure 4 illustrates Blacklight's attack detection process. For an incoming query x, Blacklight extracts its probabilistic fingerprint and stores it in the database. Blacklight runs an efficient hash match algorithm to detect overlaps between x's fingerprint and those in the database. Upon detecting sufficient overlap between x and an existing fingerprint y, it flags (x, y) as a pair of attack queries.

Key Benefits. Our fingerprint scheme has the property that any two highly similar queries will produce a near-perfect match in their fingerprints. In other words, small changes to an image are highly unlikely to impact its fingerprint. The use of secure one-way hash and probabilistic downsampling means that unless they can reverse the hashing algorithm, an adversary cannot alter an image's fingerprint without significantly altering its content (further confirmed in §9.1).

Our fingerprints also greatly reduce the storage overhead of past queries, and the computation costs of comparing queries in similarity. Specifically, the search for highly similar queries reduces down to a hash set comparison problem, which takes near-constant time in general (see §6).

Prior Work on Probabilistic Fingerprints. Probabilistic fingerprints have been used for similarity detection in text (e.g., detecting code plagiarism [7, 24, 59, 63], network intrusion and malware [54, 58, 64] and spam emails [46, 84]) and file systems [52]. The contributions of our work include i) extending probabilistic fingerprints beyond the text domain, ii) customizing its design to identify similar image queries to a DNN model (see §6), and iii) a formal analysis to model both false positives and attack detection rates and their dependency on fingerprinting parameters (see §7).

6 Detailed Design of Blacklight

We now present the detailed design of Blacklight, including preprocessing, probabilistic fingerprinting, and comparison algorithms, which together form our proposed detector. We also discuss options to mitigate attacks after detection. Note that Blacklight works as an external add-on, and requires no modifications to the DNN model.

6.1 Preprocessing: Salted Pixel Quantization

Given an incoming image query x, Blacklight first runs a quantization function on each pixel of x. This serves two purposes. First, it converts continuous pixel values into a finite set of discrete values, which are then used to compute hashes of x during fingerprinting. Second, quantization increases similarity between (attack) queries. This is particularly true for black-box attacks that iteratively optimize queries by gradually modifying every single pixel on the image [6, 14, 30] – the use of quantization effectively nullifies changes to image hashes created by these minor modifications without inducing false positives. We confirm this empirically [4] and find the hash overlap between attack queries (on `CIFAR10`) increases rapidly with the quantization step $\mathbf{q}$ to approach 100%, while those between benign queries remain low. Note that this step is used only for attack detection. If the input is considered benign, the original, unaltered query is sent to the DNN model.

Furthermore, Blacklight employs a *salted* pixel quantiza-

[4] Details in our extended version [42]

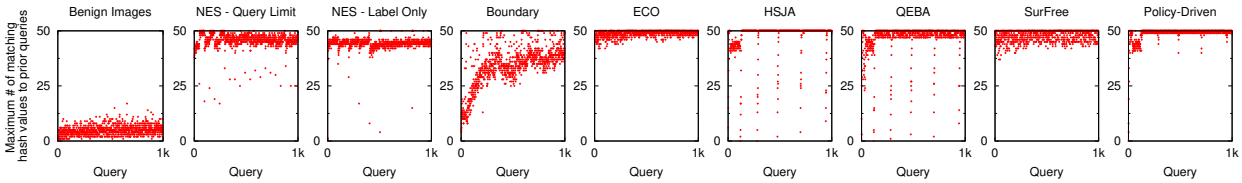

Figure 6: *We empirically show that probabilistic fingerprints preserve the query similarity in black-box attack sequences. We plot the* maximum fingerprint overlap *between x and that of any prior query in a benign query sequence (left most) and eight attack query sequences. Here the maximum matching is bounded by* $\mathbf{S} = 50$.

tion function to resist reverse engineering attacks:

$$Q(x, salt_Q, \mathbf{q}) = \lfloor \frac{(x + salt_Q) \mod 255}{\mathbf{q}} \rfloor \qquad (2)$$

where $salt_Q$ is a randomly generated salt image (of the same dimensions as x) and $\mathbf{q}$ is the quantization step (a system parameter). Here all pixel values of x and $salt_Q$ are normalized to $[0, 255]$. Later in §9 we show adding a random salt improves Blacklight's robustness against adaptive attacks.

6.2 Computing Probabilistic Fingerprints

We now describe the detailed process to compute the probabilistic fingerprint on a (quantized) query image x.

Converting an image into N segments. Blacklight first "flattens" the 2D image into a single pixel sequence by concatenating rows of pixels together; then applies a sliding window of fixed size $\mathbf{w}$ on this sequence, iteratively moving the sliding window by $\mathbf{p}$ (referred to as the sliding step). This produces $\mathbf{N} = (|x| - \mathbf{w} + \mathbf{p})/\mathbf{p}$ overlapping pixel segments, each of length $\mathbf{w}$. Any two consecutive segments overlap by $\mathbf{w} - \mathbf{p}$ pixels, and each pixel in x is included in $\mathbf{w}/\mathbf{p}$ segments.

Hashing each segment. For each segment i ($i \in [1, \mathbf{N}]$), Blacklight applies a secure one-way hash function (e.g., SHA-3 combined with a random salt value chosen by the defender) and produces a hash value h_i. This creates a full hash set $\mathbf{H}_x = (h_1, h_2, ..., h_{\mathbf{N}})$ for query x, with $\mathbf{N}$ hash entries. For example, for `CIFAR10` ($|x| = 32 \times 32 \times 3 = 3072$), $\mathbf{N} = 3053$ when $\mathbf{w} = 20$, $\mathbf{p} = 1$. An illustration of this sliding window hashing scheme is shown in Figure 5.

Selecting a subset of hashes as the fingerprint. From x's full hash set $\mathbf{H}_x$, Blacklight selects the top $\mathbf{S}$ hash values (sorted by *numerical order*) as its probabilistic fingerprint, denoted as $\mathbb{S}(\mathbf{H}_x)$. Since the output distribution of the one-way hash is random, choosing the top $\mathbf{S}$ hash values by numerical order serves as an efficient downsampling algorithm that is deterministic[5] to the defender but unpredictable to an adversary (since predicting the top $\mathbf{S}$ hash values requires predicting the full hash set).

The use of probabilistic fingerprinting puts a *hard* limit on the overhead of fingerprint storage and comparison, while preserving the high similarity among attack queries. Figure 6 shows a sample measurement on query similarity, for the eight black-box attacks discussed in §2.2. Here we measure, for each query x_i in an attack sequence, the maximum number of matching hashes between x_i's fingerprint and any of its prior queries in the same sequence. For reference, we also compute the number of matching hashes among benign images. We see that many attack queries display fingerprints highly similar to at least one prior query in the same sequence, while benign queries share minimal overlap in fingerprints. Thus Blacklight can quickly detect black-box attacks after seeing only a small number of queries.

6.3 Comparing and Matching Fingerprints

Upon receiving a new query x, Blacklight computes its fingerprint $\mathbb{S}(\mathbf{H}_x)$ and compares it to all prior fingerprints stored in the database. If any stored fingerprint shares more than $\mathbf{T}$ hash entries with $\mathbb{S}(\mathbf{H}_x)$, then x is flagged as an attack image. Here, the value of $\mathbf{T}$ can be configured to meet the desired false positive rate. Later in §7, we analytically show that by properly configuring $\mathbf{T}$ and $\mathbf{S}$, we can achieve accurate attack detection at a low false positive rate.

Computing the maximum overlap between the fingerprint of a query and n stored fingerprints is non-trivial. A simple algorithm would incur computation cost of $O(n)$. We use a better algorithm which stores a query x's fingerprint into a hashmap using each of its hash entry as a key. The maximum overlap with all n queries can be found by retrieving all queries associated with each key in x's fingerprints, and counting the max frequency of any query in that set. An efficient implementation can produce average runtime that is a constant independent of n. We leave the design and analysis of an efficient hashset matching algorithm to future work. We present detailed performance overheads in §8.6.

6.4 Mitigating Attacks after Detection

Detecting the presence of a query-based black-box attack is just a first step in protecting DNN models. A persistent attacker can simply switch accounts and/or IP addresses and continue with additional queries. Here, we discuss options for mitigation after an attack is detected.

Ban accounts. As a response, banning an account or blocking an IP address is not ideal. First, it means each false positive incurs a high penalty, which might be undesirable in

[5] Deterministic means that the downsampled hash set holds the same property of the full hash set: highly similar (quantized) queries will have highly similar fingerprints. We also verified this empirically (details in our extended version [42]).

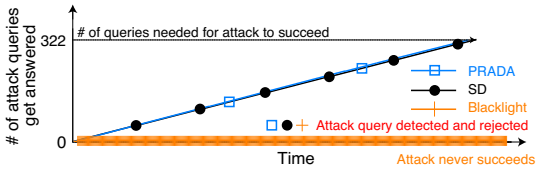

Figure 7: *By detecting/rejecting most of attack queries (regardless of account usage), Blacklight effectively resists persist attackers, which existing defenses fail to address.*

some application settings. Second, this does little to deter resource rich attackers, who can continue the attack using Sybil accounts, which are difficult to eradicate in practice.

Return misguided outputs. We also consider a more elaborate scheme where the defender intentionally misleads the attacker by returning carefully biased query outputs, perhaps towards secondary goals like identifying the attacker. This approach faces additional challenges. First, crafting biased responses requires significantly more computation and state-keeping at the defender. Second, the defender must be careful to avoid returning valid responses to actual attack queries.

Reject all detected queries. Ultimately we chose a simple strategy: reject all detected attack queries. This mitigation is effective in preventing attacks *IFF* the ratio of attack queries detected is high. If most attack queries are rejected, the attack sequence takes a very long time to converge and succeed. The benefit of this approach is that it does not rely on detecting or reducing Sybil accounts, and false positives have minimal impact on benign users.

In §8, we evaluate the impact of mitigation on persistent attackers who continue to submit attack queries after query rejection. Figure 7 provides a preview in terms of the # of attack queries got answered under Blacklight, using the persistent attack trace of Figure 2. Blacklight rejects almost all the attack queries, preventing the attack from making progress.

7 Formal Analysis

We formally examine Blacklight by modeling the process of probabilistic fingerprinting. We derive analytical bounds on $Q(\Delta)$, the probability of Blacklight flagging a query pair (x,y) as attacks, as a function of the full hash difference between the two, $\Delta = \text{diff}(\mathbf{H}_x, \mathbf{H}_y)$. We then estimate Blacklight's false positive rate and attack detection rate by $Q(\Delta_{benign})$ and $Q(\Delta_{attack})$. Here Δ_{benign} is the minimum full hash difference between benign queries and Δ_{attack} is the maximum full hash difference between attack queries. Our key results are: (i) $Q(\Delta)$ decays fast with Δ, (ii) Blacklight can detect attacks at a low false positive rate, i.e., $Q(\Delta_{attack}) \to 1$, $Q(\Delta_{benign}) \to 0$, if $\Delta_{benign} >> \Delta_{attack}$, (iii) the bound on $Q(\Delta)$ can guide the selection of Blacklight's configuration parameters ($\mathbf{w}$, $\mathbf{p}$, $\mathbf{q}$, $\mathbf{S}$ and $\mathbf{T}$). We leave the detailed analysis and proof to the extended version [42].

8 Experimental Evaluation

Using four different image classification tasks (and datasets), we empirically evaluate Blacklight against eight SOTA black-box attacks. Our experiments seek to understand 1) the effectiveness of Blacklight in both attack detection and mitigation; 2) the false positive rate under realistic settings; 3) impact of Blacklight configuration; 4) Blacklight's storage and computation cost; 5) applying Blacklight to other domain.

8.1 Experimental Setup

We apply Blacklight to protect DNN models developed for image classification. Our experiments cover a wide range of input size/content and model architectures, allowing us to evaluate Blacklight under a diverse set of conditions. We include the detailed model architectures and training configurations, attack parameters and distance metrics, and Blacklight's parameter settings in the extended version [42].

Image Classification Tasks. We consider four representative tasks: `MNIST` [39], `GTSRB` [66], `CIFAR10` [36] and `ImageNet` [60]. We summarize in Appendix §C these tasks and their associated models in Table 9.

Attack Configurations. We implement and run the eight black-box attacks list in Table 1 against each of the above four classification models. For `MNIST`, `GTSRB` and `CIFAR10`, we randomly select 1000 images from their test datasets and use each as the source image of the attack (*i.e.* x_0). For `ImageNet`, we randomly select 500 source images (due to its higher computation cost). We run each attack until it terminates (i.e., successfully generating an adversarial example) or reaches 100K queries, whichever occurs first.

When configuring each attack, we follow its original paper and use the same L_p distance metric (L_2 or L_∞) stated in the paper. Since L_2 distance depends on model input size, we use the *normalized* L_2 distance $\sqrt{\frac{1}{|x|}\sum_{i=0}^{|x|}(x_i - x'_i)^2}$.

We set the perturbation budget ε such that most attacks succeed in absence of defenses. For reference, the standard ε for white-box attacks is 0.03 for L_∞ and <0.03 for *normalized* L_2 [12]. Thus black-box attacks should use a larger budget because they are naturally harder to succeed – our experiments on the eight SOTA black-box attacks confirm that a budget of 0.03 leads to significant attack failures. Thus we increase ε=0.05 for both L_∞ and *normalized* L_2 to allow most attacks to succeed. The only exceptions are L_∞ attacks against `MNIST` since ε=0.1 is necessary for them to succeed.

Blacklight Configuration. To demonstrate the generality of Blacklight, we set these parameters to be the same default values for all four tasks, rather than "optimizing" them per task. The only exception is $\mathbf{w}$ – our default value is 20, but we increase it to 50 for `MNIST` (due to its large black background) and `ImageNet` (due to its large image size).

We choose these values following our formal analysis. In particular, we choose $\mathbf{T} = \mathbf{S}/2 = 25$ by modeling how $\mathbf{T}$ af-

Task	Attack	w. Detection			w. Mitigation	w/o Blacklight	
		Attack detect %	Detection coverage	Avg queries to detection	Attack success	Attack success	Avg # attack queries
MNIST	NES - QL	100%	99.5%	2	0%	45%	66540
	NES - LO	100%	99.0%	2	0%	1%	95973
	Boundary	100%	64.2%	18	0%	21%	85467
	ECO	100%	99.9%	2	0%	43%	52780
	HSJA	100%	98.1%	6	0%	59%	9924
	QEBA	100%	98.4%	8	0%	92%	12141
	SurFree	100%	97.9%	7	0%	84%	10034
	Policy-Driven	100%	99.0%	8	0%	74%	9538
GTSRB	NES - QL	100%	98.5%	2	0%	66%	48429
	NES - LO	100%	98.0%	3	0%	17%	83823
	Boundary	100%	64.3%	22	0%	37%	76643
	ECO	100%	100.0%	2	0%	80%	27782
	HSJA	100%	98.2%	5	0%	95%	10392
	QEBA	100%	99.5%	8	0%	99%	9832
	SurFree	100%	98.3%	8	0%	98%	9192
	Policy-Driven	100%	98.1%	5	0%	100%	13021
CIFAR10	NES - QL	100%	98.3%	2	0%	100%	12621
	NES - LO	100%	98.7%	2	0%	89%	67126
	Boundary	100%	64.4%	25	0%	95%	6082
	ECO	100%	99.4%	2	0%	89%	16887
	HSJA	100%	97.1%	7	0%	100%	1205
	QEBA	100%	96.9%	6	0%	99%	1009
	SurFree	100%	96.8%	8	0%	100%	1396
	Policy-Driven	100%	97.3%	7	0%	100%	1198
ImageNet	NES - QL	100%	99.4%	2	0%	99%	11201
	NES - LO	100%	98.2%	2	0%	20%	63492
	Boundary	100%	95.1%	42	0%	74%	67356
	ECO	100%	99.6%	2	0%	93%	11304
	HSJA	100%	98.7%	7	0%	99%	12402
	QEBA	100%	98.3%	6	0%	100%	10293
	SurFree	100%	97.6%	7	0%	100%	8783
	Policy-Driven	100%	99.1%	8	0%	100%	10368

Table 2: *Blacklight's detection and mitigation results. In the last two columns, we included attack performance in absence of Blacklight: attack success rate and average attack queries required to complete an attack.*

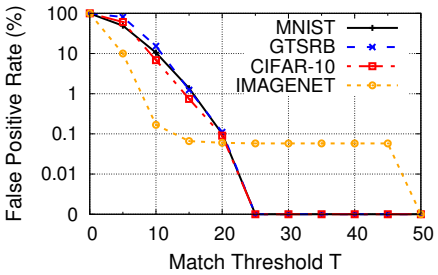

Figure 8: *Blacklight's false positive rate when fixing* $\mathbf{S} = 50$ *and varying* $\mathbf{T}$.

Label	# of Filtered	FPR	Label	# of Filtered	FPR
balloon	953	0.07%	packet	1158	0.21%
boathouse	1572	0.73%	peacock	556	0.68%
daisy	656	0.10%	pier	309	0.07%
fly	188	0.03%	rifle	905	0.48%
geyser	896	0.11%	snail	350	1.01%
hay	1192	0.79%	swing	510	0.48%
knot	817	0.14%	teapot	1715	0.14%
menu	1232	0.37%	tiger cat	1315	0.28%
mortar	1229	0.38%	toaster	3298	0.37%
nail	1696	0.83%	vault	182	0.04%

Table 3: *Blacklight's false positives on benign images crawled from Flickr. "# of Filtered" is # of images that are duplicated and have the same hash value with prior queries; "FPR" is the false positive rate per label. For each label, we run Blacklight on 80,000 Flickr images (crawled via this label).*

fects false positive and detection coverage. Figure 8 shows the measured false positive rates when varying **T**, confirming that **T**=25 achieves less than 0.1% false positive for all four tasks. In §8.5, we further explore the impact of parameter configuration by varying **w**, **p**, **q** and **S**.

Evaluation Metrics. We use the following metrics to quantify the effectiveness and cost of Blacklight.

- **False positive rate**: % of benign queries detected as attack.
- **Attack detection rate:** % of black-box attacks detected before the attack completes.
- **Detection coverage:** % of queries in an attack's query sequence identified as attack queries.
- **Avg # of queries to detection:** Average # of attack queries accepted (thus answered) before detecting an attack query.
- **Attack success rate w. mitigation:** Success rate of a persistent attack when all detected attack queries are rejected.
- **Detection overhead:** Run-time latency and storage costs.

8.2 Attack Detection and Mitigation

We evaluate Blacklight's detection rate by implementing and performing each of the eight black-box attacks against each classification model. For each attack and task combination, we run 1000 instances of the attack (500 for `ImageNet`). Each attack instance selects a random image from the test dataset as source image of the attack (x_0), and a random incorrect label as the misclassification target label.

The results for all attacks are listed in Table 2. As reference, the last two columns report the performance of these attacks *without* the Blacklight defense, in terms of attack success rate and the speed of convergence (# of queries before successfully producing an adversarial example). We see that recent attacks, especially HSJA, QEBA, SurFree, Policy-Driven, are highly successful in absence of Blacklight. Boundary and NES-LO take the longest time to converge. Some attack instances do fail to converge even after generating 100k queries (e.g., less than 50% of NES-LO complete in 100K queries for MNIST, GTSRB and ImageNet). Most of them remain unsuccessful even when increasing the query bound to 300k. Overall, a successful attack takes several thousands to tens of thousands of queries to complete.

Next, we summarize key results on Blacklight's attack detection (as shown by column 3-5 in Table 2). We see that the attack detection rate remains 100% for all attack instances, indicating that Blacklight detects *all* attacks on all models in progress. The detection coverage is also extremely high – Blacklight detects more than 96% of all attack queries, except on the Boundary attack. Another key observation is that Blacklight detects a new attack instance very quickly, often after a handful of 2–8 queries (again, more queries required for Boundary because it converges slower). In all cases, Blacklight detects an attack in less than 1% of the av-

erage number of queries required to complete the attack.

Blacklight detects Boundary slower than others. This is because Boundary advances slower in shrinking perturbation towards the L_p ball of the target, thus Blacklight detects them at a "later" stage with 100% detection rate. The three improved versions of Boundary (HSJA, QEBA, Policy) converge faster, thus Blacklight detects them faster. To further evaluate the slower Boundary attack, we run the attack for 1 million queries. We find Blacklight continues to detect (and reject) attack queries in this longer sequence, leaving the attacker with 0% success (for all four tasks). The detailed results are listed in Table 10 in Appendix.

Finally, column 6 in Table 2 reports the attack success rate when Blacklight rejects queries identified as attack queries. We see that none (0%) of persistent attackers manage to complete their attack within 100K queries. Blacklight's mitigation is highly effective because it is able to detect nearly all attack queries. Rejecting these queries prevents the attacker from making forward progress in probing model classification boundaries. This confirms that a high detection coverage is critical to defend against query-based black-box attacks.

Key Takeaways. Our results against eight SOTA black-box attacks show that Blacklight detects all attacks on all models, detects the overwhelming majority of queries in the attack sequence, and detects the attack quickly (usually in less than 8 queries, with the exception of the slow converging Boundary attack). Furthermore, by rejecting all detected attack queries, Blacklight's mitigation module ensures no attacks can complete (at least in 100K queries) for all our tested attacks and target DNN models.

Comparison to Existing Defenses. As reference, we show the performance of SD and PRADA in Table 8, using the same attack experiments described above. As discussed in §4, SD and PRADA are not designed to stop persistent attackers who switch account to continue attack. Results in Table 8 confirm this and their low detection coverage (0.8%-2.1%).

8.3 Detecting Universal Patch Attacks

We evaluate Blacklight against the only known query-based universal patch attack, Sparse-RS [20]. Table 11 in Appendix shows that Blacklight is highly effective in detecting Sparse-RS (100% detection success rate and $>$ 97.6% detection coverage). Since query-based universal patch attacks are emerging, additional work is required to thoroughly evaluate the robustness of Blacklight against them.

8.4 False Positives in Real World Settings

Since Blacklight relies on a similarity detection algorithm to detect attacks, one might wonder if duplicates or near-duplicates of images will trigger false positives. Figure 8 reports its false positives between distinctive inputs. But what about "naturally" similar images, such as different versions of the same image, or closeby frames of the same video?

We begin with a simple test to confirm that naturally occurring false positives are very low in large image repositories like ImageNet. We turn off database resets, randomly sample 1 million images from ImageNet training data, send them as queries to Blacklight, and observe a very low false positive rate of 0.37%.

False Positives in Similar Images. Next, we look at similar images of the same objects, e.g. inputs that should classify to the same labels. We crawl a large number of public real world images from Flickr [1] using keyword search on their public API. We pick 20 random labels from ImageNet, and use each as a search keyword to crawl 80,000 images for that label. We filter out images that are perfectly identical at the pixel level (we found an average of 1036 ± 696 duplicate images per label). We then take each label, and run our 80,000 images as queries to Blacklight. Even across Flickr images labeled with the same keyword, Blacklight produces a very low false positive rate of $0.37\% \pm 0.29$ over 20 labels. Detailed results for all labels are shown in Table 3.

False Positives in Video Frames. Finally, we consider the scenario where the system might receive benign queries that are highly similar by nature, e.g. image stills taken from video frames. We explore how Blacklight responds under such scenarios by testing it for false positives on the YouTube Faces dataset [74]. YouTube Faces is a collection of 3,425 videos of 1,595 different people, designed for studying unconstrained facial recognition. We use common image extraction techniques [72] to extract 587,137 video frame images from videos for 1,283 celebrities. Of these, we filter out 33,227 images that are pixel-level identical to other images, and send the remaining video frames to Blacklight. The result is a false positive rate of 1.74%. Even if Blacklight takes over half million queries per reset cycle for the highly similar queries, the false positive rate is still very low.

8.5 Impact of Parameter Configuration

We leverage our formal analysis[6] of Blacklight to configure its five system parameters: **w**, **p**, **q**, **S**, and **T**. Earlier in Figure 8 we show empirically how Blacklight's false positive rate varies with **T** and verify our strategy on configuring **T**. In the following, we study the impact of the other four parameters by testing Blacklight against the same set of attacks while varying each of these parameters. We report the false positive rate and detection coverage since the attack detection rate is always 100%. The detailed results are listed in Figure 10 in Appendix.

We summarize the key findings below. First, we confirm that **q** is a critical parameter for Blacklight – the detection coverage increases quickly as **q** goes from 1 (no quantization) to 50 (the default value) and stabilizes after that (except for Boundary). When **q** approaches 100, we start to see visible increase in false positives (>0.1%). Second, as expected,

[6] Details in our extended version [42]

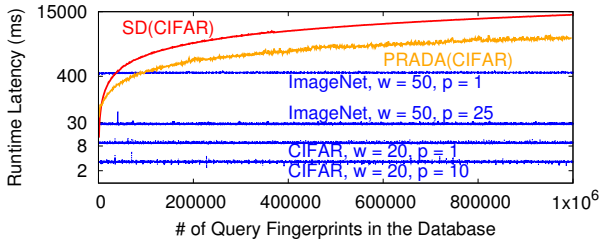

Figure 9: *Blacklight's runtime latency vs. n. Note the log Y axis. We include latency of SD and PRADA for reference.*

the sliding window size **w** is negatively correlated to false positive rate and detection coverage, while the sliding step **p** has little impact (note that $\mathbf{p} < \mathbf{w}$). Thus Blacklight should select **w** as a small value to meet the desired false positive rate. Finally, as expected **S** should be small to reduce complexity but not too small (e.g., <20) to introduce visible false positives. Overall, these results confirm our proposed theory-guided principle for choosing Blacklight's parameters.

8.6 Overhead of Blacklight

Storage. Blacklight requires a database to store fingerprints of prior queries. Our probabilistic fingerprints are extremely small. Across all of our experiments, a fingerprint is $\leq 32 \cdot \mathbf{S}$ bytes and 1.6KB for the default configuration. A database of 1 million queries only requires 2GB storage, a "negligible" value for modern servers.

Runtime. Blacklight's per-query runtime includes latency to generate the fingerprint from a query and latency to lookup the fingerprint in the query database. The former depends on the image size and the parameters (**w**, **p**) and the latter depends on the size of query database n. We configure Blacklight to its default configuration and explore the impact of sliding step **p** (i.e., increasing **p** from 1 to 10 or 25 to speed up hash computation) and the query database size n. We run Blacklight on an Intel i7 desktop server with 64 GB memory, and report the per-query runtime for two types of query images (32×32,`CIFAR10`) and (224×224, `ImageNet`) in Figure 9 as a function of n. The curves remain flat over n, suggesting that Blacklight's detection cost is independent of n. More specifically, a `CIFAR10` model inference takes 50ms (on a Nvidia Titan RTX) while Blacklight (on Intel i7) takes 4-8ms (8%-16% over 50ms) for n=1 million queries.

As reference, we compute the runtime of SD and PRADA on the same Intel i7 server, putting all n queries into a single account. They only run on `CIFAR10`, which we report in Figure 9. The latencies scale linearly with n (note the log Y axis). For n=1 million queries, SD and PRADA take 12s and 3.4s per query (24,000% and 6,800% over inference).

Further optimization. Blacklight's per query latency is dominated by the sliding window-based hash computation (99% of total runtime). We further optimize this computation using GPUs. A modified version of Blacklight running a Nvidia Titan RTX reduces per-query latency by 20x, to 0.4ms for `CIFAR10` and 20ms for `ImageNet`, almost "negligible" compared to the inference latency.

Attack	w. Detection			w. Mitigation	w/o Blacklight	
	Attack detect %	Detection coverage	Avg queries to detection	Attack success	Attack success	Avg # attack queries
TextBugger [43]	100%	99.7%	2	0%	86.0%	537
TextFooler [32]	100%	99.7%	2	0%	100.0%	669
Hard Label [50]	100%	99.9%	2	0%	100.0%	4642

Table 4: *Blacklight's detection and mitigation results on query-based black-box attacks for text classification.*

8.7 Blacklight for Text Classification

Blacklight should in principle extend to other domains where black-box adversarial attacks produce highly similar queries in the input space. The domain-specific design task is how to generate query fingerprints to enable efficient and accurate detection. Below, we show an initial Blacklight design for text classification, a critical task in NLP. DNN-based text classification is shown to be vulnerable to query-based black-box attacks [26, 32, 43, 50], with three SOTA attacks: TextFooler [32], TextBugger [43] and HardLabel [50].

Fingerprinting a sentence. The input to a text classifier is a sentence, from which Blacklight produces a fingerprint. First, we convert the sentence into an array by replacing each word with its word embedding. We quantize the array, apply a sliding window to move through the quantized array and compute hashes, and select the top **S** hashes as the query fingerprint. The parameter choices are listed in Table **??** for `IMDB` text queries. **S** and w are smaller since text sentences create "shorter" arrays, while **T** remains $\mathbf{S}/2$.

Blacklight performance. We run Blacklight on the three SOTA attacks on the `IMDB` dataset [48]. The results in Table 4 show that Blacklight achieves 100% detection rate and >99.7% detection coverage, only takes 2 queries to detect an attack (and reject the second query). As such, no attack ever succeeds. For all of these tests, the false positive rate is only 0.49%. Overall, these results offer clear evidence that Blacklight can potentially generalize to other domains using the same probabilistic fingerprint methodology.

9 Adaptive Attacks

A meaningful defense must be robust against adaptive countermeasures from attackers with full knowledge of the defense. We explored a number of customized adaptive attacks against Blacklight, and present the strongest countermeasures, organized into three groups: 1) reducing query similarity for attack sequences, 2) reducing queries needed for successful attacks and 3) leveraging resets in Blacklight. Given the similarity between the attacks, we only apply countermeasures to 5 of 8 attacks: NES (QL & LO), Boundary, ECO and HSJA.

Attack Type	Default $\mathbf{T} = 25$ (FPR = 0.0%)				$\mathbf{T} = 15$ (FPR = 0.74%)			
	0.05	0.1	0.15	0.2	0.05	0.1	0.15	0.2
NES - QL	100%	100%	100%	100%	100%	100%	100%	100%
NES - LO	100%	100%	100%	100%	100%	100%	100%	100%
Boundary	100%	100%	75%	40%	100%	100%	100%	95%
ECO	100%	100%	100%	100%	100%	100%	100%	100%
HSJA	100%	100%	55%	40%	100%	100%	80%	40%

Table 5: *Blacklight detection rate for attacks using larger perturbation budgets (0.1-0.2) for* `CIFAR10`. *Lowering* $\mathbf{T}$ *largely improves detection when attackers operate on very large perturbations, with a small increase in false positives.*

9.1 Reducing Query Similarity

With knowledge of how Blacklight works, the straightforward adaptive attack is to evade detection by reducing similarity between attack queries. Below we present four types of adaptive attacks that add perturbations to attack queries to reduce similarity between them.

Evasion via Image Transformations. An attacker can try to evade detection by adding additional perturbations to attack queries, where ideally these perturbations do not disrupt the iterative optimization process, but are significant enough to make fingerprints of attack queries different. We explore two types of image transformations: 1) adding Gaussian noise, and 2) applying image augmentation like shift, rotation, zoom and blending. We apply these transformations to attack queries and send them to Blacklight. We first examine how these transformations affect the attack in absence of Blacklight, and confirm that they do introduce different levels of disruptions (none to 100%). On the other hand, for all the transformed attack sequences that will lead to a successful attack in absence of Blacklight, Blacklight detects all of them, i.e., 100% attack detection rate. Further details are in Appendix§ E and Table 12.

Increasing Learning Rates. The attacker can also try to increase dissimilarity between consecutive queries by tweaking their learning rate parameter. Learning rate controls the difference between two adjacent queries when estimating gradients. This does not apply to gradient estimation free attacks (Boundary and ECO). We only explore different learning rate for NES-QL, NES-LO and HSJA attacks. For two variants of NES, we gradually increase learning rate more than 1000 fold. While the attack success rate drops to 0%, detection success rate remains 100%. For HSJA, we gradually grow learning rate up to a factor of 10^6, until changes in learning rate no longer impact gradient estimation results. Here, attack success rate steadily drops (eventually to 15%), but detection remains at 100% throughout.

Increasing Perturbation Budgets. Our evaluation so far assumes the attacker's perturbation budget is limited to commonly accepted values: 0.05 for both L_∞ and *normalized* L_2. Future attacks might tolerate a higher perturbation budget in specific settings. Thus, we evaluate Blacklight's detection performance against attacks on `CIFAR10` with larger perturbation budgets. For all attacks, we incrementally increase the budget from 0.05 all the way up to 0.2, and measure Blacklight's attack detection rates for each budget level (running 20 attack instances for each data point). We show that the attack detection rates for NES variants and ECO remain steady at 100%, but Boundary and HSJA begin to evade detection when *normalized* $L_2 = 0.15$ (or $L_2 = 8.3$).

Blacklight can compensate by tuning the fingerprint matching threshold $\mathbf{T}$. Table 5 shows that by lowering $\mathbf{T}$ from 25 to 15, we can dramatically increase detection rates, restoring perfect detection to most attacks (except HSJA at *normalized* $L_2 = 0.15/0.2$ ($L_2 = 8.3/11.1$) and Boundary at *normalized* $L_2 = 0.2$ ($L_2 = 11.1$)). This drop in $\mathbf{T}$ only increases false positive rates by 0.74%.

We further validate our results on the other three tasks for the two fastest converging attacks (ECO and HSJA) and the results (Table 13) are consistent with `CIFAR10`. Finally, we also perform analysis on the L_2 distances between benign images to provide a baseline for reasonable L_2 budget for adversarial attacks in Appendix §E.

Evasion via Guided Transformation. Beyond first order adaptive attacks, we worked hard to design more powerful attacks specifically targeting Blacklight. Assuming a Blacklight system's parameters $\mathbf{q}$ and $\mathbf{w}$ are unknown to an attacker, the strongest attack we could design is the two-pronged reverse engineer attack, where an attacker first uses queries to probe the limits of $\mathbf{q}$ and $\mathbf{w}$, and then leverages those results to optimize a guided transformation attack.

The high-level intuition is that an attacker can optimally spread out their perturbation budget across the image, if they understand Blacklight and learned its specific configuration parameters. As long as there is at least one pixel changed (after pixel quantization) for some sliding window, hash values of the window will be changed. Thus, the attacker just needs to make sure that for each window, at least one pixel is different from all prior queries after quantization. In this case, Blacklight's use of $salt_Q$ in eq (2) is crucial to resisting these guided transformation attacks. Next, we summarize the attack and results when Blacklight turns $salt_Q$ off or on.

(i) Guided transformation (Blacklight's $salt_Q$ off). An attacker begins by estimating quantization step $\mathbf{q}$ and using it to compute quantization boundary B, followed by estimating value of $\mathbf{w}$. It does this by issuing pairs of queries with a minimal perturbation based on an initial estimate of $\mathbf{q}$ or $\mathbf{w}$, and observing whether the second query is detected as an attack. This is repeated using binary search until both $\mathbf{q}$ and $\mathbf{w}$ are determined. Finally, the attacker computes B from $\mathbf{q}$, and then the optimal layout of modified pixels to maximize the number of substring windows affected by the perturbation. The attacker uses this process to modify each query to evade detection while iteratively optimizing queries to generate the adversarial example. We implement this attack on top of the two fastest converging attacks (ECO and HSJA) and the slowest attack (Boundary). Table 6 shows that the attacker achieves no more than 25% success rate for all tasks.

Task	Blacklight's $salt_Q$ off			$salt_Q$ on			attacker knows $(\mathbf{p}, \mathbf{q}, \mathbf{w})$, $salt_Q$ on		
	Boundary	ECO	HSJA	Boundary	ECO	HSJA	Boundary	ECO	HSJA
MNIST	0%	0%	0%	0%	0%	0%	0%	0%	0%
GTSRB	10%	5%	5%	0%	0%	0%	0%	0%	0%
CIFAR10	20%	15%	25%	0%	0%	0%	0%	0%	0%
ImageNet	5%	10%	20%	0%	0%	0%	0%	0%	0%

Table 6: *Attack success rate using guided transformations attacks.*

Metrics	NES	AutoZOOM
Attack success %	100%	100%
Attack detect %	100%	100%
Detection coverage	99.1%	98.9%
Avg queries to detection	2	2
Avg # of attack queries	1473	1240

Table 7: *Blacklight vs. hybrid batch attacks.*

(ii) Guided transformations (Blacklight's $salt_Q$ on). The defender can overcome the above adversary by making it harder to extract the quantization boundary. Blacklight does so by adding a "salt" to the quantization process, i.e., $salt_Q$ in eq. (2). This defeats attempts by the attacker to reverse engineer $\mathbf{q}$ and B. Without knowledge of $\mathbf{q}$, an attacker can still launch a weaker version of the attack, but must overshoot on perturbation to increase chances of it persisting through the salted quantization and alter the hashes. We implement such attack by altering 5, 10, and 15 out of every 20 pixels within the perturbation budget. When applying this new attack on top of ECO, HSJA, and Boundary, the attacker still achieves 0% success on all tasks, while Blacklight maintains a high detection coverage (78%). This confirms the significant robustness gained by adding the salt.

Guided Transformations when Attacker Knows $(\mathbf{q}, \mathbf{p}, \mathbf{w})$. Finally, we consider the strongest guided transformation attack – the attacker knows the exact values of $\mathbf{q}$, $\mathbf{p}$, $\mathbf{w}$ and can better perturb queries to evade detection.

To make a query x evade detection, the attacker must ensure that for each window, at least one pixel of x is different from all prior queries after quantization. This is because Blacklight's one-way hash distribution and the top $\mathbf{S}$ hash choices remain unpredictable to the attacker. Knowing $\mathbf{q}$, $\mathbf{p}$, $\mathbf{w}$ helps the attacker to optimize the pixel perturbation. For example, now in each window changing a pixel by $\mathbf{q}$ or $-\mathbf{q}$ will change the hash despite the use of $saltQ$. To make x's full hashes different from those of all prior attack queries, we apply a permutation-based pixel selection algorithm to minimize the total perturbation (see Algorithm1 in Appendix).

Even with this strong attack, attackers still have 0% success rate after sending 100K queries (see Table 6). These attack queries do bypass Blacklight's detection, but the attack's iteration optimization process never converges to generate an adversarial example (regardless of the perturbation budget). This is because the perturbation applied to individual attack queries in order to evade detection is too large to make the query results useful for attack optimization, i.e., they fail to capture detailed decision boundaries of the target model. As such, the iterative optimization process fails to make concrete progress but "randomly" wanders around.

Summary. Together, our experiments with guided transformation attacks show that (1) salted quantization is important to resist advanced attackers, and (2) under the Blacklight defense, attackers now face two conflicting goals when building attack queries: evading Blacklight's detection or advancing the attack's iterative optimization process using queries.

9.2 Reducing Number of Attack Queries

Another way to evade Blacklight is to reduce the queries needed for an attack to succeed. Since Blacklight examines similarity between a new query and past queries, the fewer the queries needed, the lower the probability that the attack query will be detected. We explore two adaptive attacks that focus on reducing attack queries needed.

Hybrid Black-Box Attacks. Substitute model based priors can be useful for planning attack queries [19, 29, 33, 67]. For example, adversarial examples generated from a substitute model can serve as a good starting point to launch query-based black-box attacks, allowing the attacker to use a smaller number of queries to complete the attack [67]. We run two of these hybrid attacks [67] (NES and AutoZOOM) while using Blacklight to protect the target model. For each attack, we run 100 attack sequences on CIFAR10 and report our results in Table 7. We see that the two hybrid attacks do reduce the number of queries required for complete an attack, Blacklight still leads to 100% attack detection, 99% of detection coverage, and detect attack queries after just 2 queries.

Optimal Black-Box Attacks. Since black-box attacks are continuously evolving in query efficiency, we also evaluate Blacklight against two types of highly efficient attacks that are possible but do not yet exist. First, we consider extremely "query-efficient" black-box attacks that require orders of magnitude fewer attack queries than current attacks by downsampling existing attack sequences. We find that even when attacks are able to complete in 500, 100, or 50 queries, Blacklight still detects them near perfectly (100% detection rate for 4 attacks and 89% for Boundary attack).

Second, we imagine a "perfect-gradient" black-box algorithm that is somehow able to perfectly predict gradient functions from the results of its attack queries, as accurately as a white-box attack. Our results show Blacklight detects 100% of attacks driven by CW [12], and 81% of attacks driven by PGD [49]. The details are listed in Table 14, Appendix §E.

9.3 Evasion by Exploiting Reset Window

Finally, to guarantee the efficacy of Blacklight, the defender would reset the system periodically. Thus, a patient attacker can leverage the reset feature to evade detection.

Pause and Resume Attacks. Adversaries can try to evade detection by exploiting the fact that Blacklight periodically

resets its database to remove all fingerprints. They can pause their attack every time it receives a rejection response, and resuming the attack the next time Blacklight resets its database. We experiment on all five black-box attacks using this strategy against a `CIFAR10` model and Blacklight. We run 100 instances of each attack, and show average total queries needed for each attack to succeed, and the average number of reset cycles that requires in Table 15. If we reset Blacklight every 24 hours, the fastest successful attacker would complete an attack (using HSJA) in 1092 days or roughly 3 years. While this strategy does allow for a successful attack, the time cost to perform this attack makes it highly impractical.

10 Conclusion and Limitations

Blacklight protects DNN models against query-based black-box attacks, using a probabilistic fingerprint to detect highly similar queries generated by attack optimization. Blacklight achieves near-perfect detection against eight SOTA attacks with negligible false positives, resists persistent attackers, and is robust to a range of adaptive and even idealized countermeasures. We also demonstrated that Blacklight can successfully generalize to some text classification tasks.

Blacklight faces two limitations that demand further research. First, it is unable to defend against substitute model (SM) attacks, but can be combined with SM defenses to launch a more complete defense against both types of black-box attacks (see Appendix §A for initial results). Second, Blacklight relies on the fact that *existing* query-based black-box attacks all produce highly similar queries during their iterative optimization process, a phenomenon rarely seen in benign queries. It is not future-proof, i.e. a (future) attack breaking this assumption would evade Blacklight.

Acknowledgements

We thank our anonymous reviewers for their insightful feedback. This work is supported in part by NSF grants CNS-1949650, CNS-1923778, CNS-1705042, by C3.ai DTI, and by the DARPA GARD program. Huiying Li is supported in part by a Facebook Fellowship and a Siebel Scholarship. Emily Wenger is also supported by a GFSD Fellowship and a Harvey Fellowship. Both Emily Wenger and Shawn Shan are also supported by Neubauer Fellowships at the University of Chicago. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of any funding agencies.

References

[1] Flickr, 2020. http://www.flickr.com/.

[2] Photodna, 2021. https://www.microsoft.com/en-us/photodna.

[3] Maksym Andriushchenko et al. Square attack: a query-efficient black-box adversarial attack via random search. In *Proc. of ECCV*, 2020.

[4] Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *Proc. of ICML*, 2018.

[5] Arjun Nitin Bhagoji, Warren He, Bo Li, and Dawn Song. Practical black-box attacks on deep neural networks using efficient query mechanisms. In *Proc. of ECCV*, 2018.

[6] Wieland Brendel, Jonas Rauber, and Matthias Bethge. Decision-based adversarial attacks: Reliable attacks against black-box machine learning models. In *Proc. of ICLR*, 2018.

[7] Sergey Brin, James Davis, and Hector Garcia-Molina. Copy detection mechanisms for digital documents. In *Proc. of SIGMOD*, 1995.

[8] Jacob Buckman, Aurko Roy, Colin Raffel, and Ian Goodfellow. Thermometer encoding: One hot way to resist adversarial examples. In *Proc. of ICLR*, 2018.

[9] Nicholas Carlini and David Wagner. Defensive distillation is not robust to adversarial examples. *arXiv:1607.04311*, 2016.

[10] Nicholas Carlini and David Wagner. Adversarial examples are not easily detected: Bypassing ten detection methods. *Proc. of AISec*, 2017.

[11] Nicholas Carlini and David Wagner. Magnet and efficient defenses against adversarial attacks are not robust to adversarial examples. *arXiv:1711.08478*, 2017.

[12] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *Proc. of IEEE S&P*, 2017.

[13] Casey Chan. What facebook deals with everyday: 2.7 billion likes, 300 million photos uploaded and 500 terabytes of data, 2012. https://gizmodo.com/what-facebook-deals-with-everyday-2-7-billion-likes-3-5937143.

[14] Jianbo Chen, Michael I. Jordan, and Martin J. Wainwright. Hopskipjumpattack: A query-efficient decision-based attack. In *Proc. of IEEE S&P*, 2020.

[15] Pin-Yu Chen et al. Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In *Proc. of AISec*, 2017.

[16] Pin-Yu Chen et al. Ead: elastic-net attacks to deep neural networks via adversarial examples. In *Proc. of AAAI*, 2018.

[17] Steven Chen, Nicholas Carlini, and David Wagner. Stateful detection of black-box adversarial attacks. In *Proc. of ACM SPAI*, 2020.

[18] Minhao Cheng et al. Sign-opt: A query-efficient hard-label adversarial attack. In *Proc. of ICLR*, 2019.

[19] Shuyu Cheng et al. Improving black-box adversarial attacks with a transfer-based prior. In *Proc. of NeurIPS*, 2019.

[20] Francesco Croce et al. Sparse-rs: a versatile framework for query-efficient sparse black-box adversarial attacks. *arXiv:2006.12834*, 2020.

[21] G. S. Dhillon et al. Stochastic activation pruning for robust adversarial defense. In *Proc. of ICLR*, 2018.

[22] Yinpeng Dong, Tianyu Pang, Hang Su, and Jun Zhu. Evading defenses to transferable adversarial examples by translation-invariant attacks. In *Proc. of CVPR*, 2019.

[23] John R. Douceur. The Sybil attack. In *Proc. of IPTPS*, 2002.

[24] Stéphane Ducasse, Matthias Rieger, and Serge Demeyer. A language independent approach for detecting duplicated code. In *Proc. of ICSM*, 1999.

[25] Ryan Feng et al. Query-efficient physical hard-label attacks on deep learning visual classification. *arXiv preprint arXiv:2002.07088*, 2020.

[26] Ji Gao, Jack Lanchantin, Mary Lou Soffa, and Yanjun Qi. Black-box generation of adversarial text sequences to evade deep learning classifiers. In *Proc. of IEEE S&P Workshops*. IEEE, 2018.

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

[28] Warren He et al. Adversarial example defenses: Ensembles of weak defenses are not strong. In *Proc. of WOOT*, 2017.

[29] Zhichao Huang and Tong Zhang. Black-box adversarial attack with transferable model-based embedding. In *Proc. of ICLR*, 2019.

[30] Andrew Ilyas, Logan Engstrom, Anish Athalye, and Jessy Lin. Black-box adversarial attacks with limited queries and information. In *Proc. of ICML*, 2018.

[31] Nathan Inkawhich et al. Perturbing across the feature hierarchy to improve standard and strict blackbox attack transferability. *Proc. of NeurIPS*, 2020.

[32] Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. Is bert really robust? a strong baseline for natural language attack on text classification and entailment. In *Proc. of AAAI*, 2020.

[33] Mika Juuti, Buse Gul Atli, and N Asokan. Making targeted black-box evasion attacks effective and efficient. In *Proc. of AISec*, 2019.

[34] Mika Juuti, Sebastian Szyller, Samuel Marchal, and N Asokan. Prada: protecting against dnn model stealing attacks. In *Proc. of EuroS&P*, 2019.

[35] Neal Krawetz. Kind of like that, 2013. http://www.hackerfactor.com/blog/index.php?/archives/529-Kind-of-Like-That.html.

[36] Alex Krizhevsky et al. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

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

[38] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial machine learning at scale. *Proc. of ICLR*, 2017.

[39] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proc. of the IEEE*, 1998.

[40] Nicole Lee. Having multiple online identities is more normal than you think. Engadget, March 2016. https://www.engadget.com/2016/03/04/multiple-online-identities.

[41] Huichen Li et al. Qeba: Query-efficient boundary-based blackbox attack. In *Proc. of CVPR*, 2020.

[42] Huiying Li et al. Blacklight: Scalable defense for neural networks against query-based black-box attacks. *arXiv:2006.14042*, 2022.

[43] J Li et al. Textbugger: Generating adversarial text against real-world applications. In *Proc. of NDSS*, 2019.

[44] Jiadong Lin et al. Nesterov accelerated gradient and scale invariance for adversarial attacks. In *Proc. of ICLR*, 2019.

[45] Yanpei Liu, Xinyun Chen, Chang Liu, and Dawn Song. Delving into transferable adversarial examples and black-box attacks. In *Proc. of ICLR*, 2017.

[46] Zhijun Liu et al. Detecting and filtering instant messaging spam-a global and personalized approach. In *Proc. of NPSec*, 2005.

[47] Xingjun Ma et al. Characterizing adversarial subspaces using local intrinsic dimensionality. In *Proc. of ICLR*, 2018.

[48] Andrew Maas et al. Learning word vectors for sentiment analysis. In *Proc. of ACL:HLT*, 2011.

[49] Aleksander Madry et al. Towards deep learning models resistant to adversarial attacks. *arXiv:1706.06083*, 2017.

[50] Rishabh Maheshwary, Saket Maheshwary, and Vikram Pudi. Generating natural language attacks in a hard label black box setting. In *Proc. of AAAI*, 2021.

[51] Thibault Maho, Teddy Furon, and Erwan Le Merrer. Surfree: a fast surrogate-free black-box attack. *arXiv:2011.12807*, 2020.

[52] Udi Manber. Finding similar files in a large file system. In *Proc. of USENIX Winter Technical Conference*, volume 94, pages 1–10, 1994.

[53] Seungyong Moon, Gaon An, and Hyun Oh Song. Parsimonious black-box adversarial attacks via efficient combinatorial optimization. In *Proc. of ICML*, 2019.

[54] Ciprian Oprişa, George Cabău, and Gheorghe Sebestyen Pal. Malware clustering using suffix trees. *Journal of Computer Virology and Hacking Techniques*, 2016.

[55] Nicolas Papernot et al. Distillation as a defense to adversarial perturbations against deep neural networks. In *Proc. of IEEE S&P*, 2016.

[56] Nicolas Papernot et al. Practical black-box attacks against machine learning. In *Proc. of ACM AsiaCCS*, 2017.

[57] Nicolas Papernot, Patrick McDaniel, and Ian Goodfellow. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *arXiv:1605.07277*, 2016.

[58] Vassil Roussev. Hashing and data fingerprinting in digital forensics. *IEEE Security & Privacy*, 2009.

[59] Chanchal K Roy, James R Cordy, and Rainer Koschke. Comparison and evaluation of code clone detection techniques and tools: A qualitative approach. *Science of computer programming*, 2009.

[60] Olga Russakovsky et al. ImageNet Large Scale Visual Recognition Challenge. *Proc. of IJCV*, 2015.

[61] P. Samangouei, M. Kabkab, and R. Chellappa. Defensegan: Protecting classifiers against adversarial attacks using generative models. In *Proc. of ICLR*, 2018.

[62] Shawn Shan, Emily Wenger, Bolun Wang, Bo Li, Haitao Zheng, and Ben Y. Zhao. Gotta catch 'em all: Using honeypots to catch adversarial attacks on neural networks. In *Proc. of CCS*, 2020.

[63] Narayanan Shivakumar and Hector Garcia-Molina. Scam: A copy detection mechanism for digital documents. In *Proc. of ACM DL*, 1995.

[64] Sumeet Singh, Cristian Estan, George Varghese, and Stefan Savage. Automated worm fingerprinting. In *Proc. of OSDI*, 2004.

[65] Y. Song et al. Pixeldefend: Leveraging generative models to understand and defend against adversarial examples. In *Proc. of ICLR*, 2018.

[66] Johannes Stallkamp, Marc Schlipsing, Jan Salmen, and Christian Igel. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural Networks*, 2012.

[67] Fnu Suya, Jianfeng Chi, David Evans, and Yuan Tian. Hybrid batch attacks: Finding black-box adversarial examples with limited queries. In *Proc. of USENIX Security*, 2020.

[68] Florian Tramèr et al. Ensemble adversarial training: Attacks and defenses. In *Proc. of ICLR*, 2018.

[69] Chun-Chen Tu et al. Autozoom: Autoencoder-based zeroth order optimization method for attacking black-box neural networks. In *Proc. of AAAI*, 2019.

[70] Twitter. Twitter turns six, 2012. https://blog.twitter.com/official/en_us/a/2012/twitter-turns-six.html.

[71] Jonathan Uesato, Brendan O'donoghue, Pushmeet Kohli, and Aaron Oord. Adversarial risk and the dangers of evaluating against weak attacks. *arXiv:1802.05666*, 2018.

[72] Bolun Wang, Yuanshun Yao, Shawn Shan, Huiying Li, Bimal Viswanath, Haitao Zheng, and Ben Y. Zhao. Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In *Proc. of IEEE S&P*, 2019.

[73] Wierstra et al. Natural evolution strategies. In *Proc. of IEEE World Congress on Computational Intelligence*, 2008.

[74] Lior Wolf, Tal Hassner, and Itay Maoz. Face recognition in unconstrained videos with matched background similarity. In *Proc. of CVPR*, 2011.

[75] Eric Wong, Leslie Rice, and J Zico Kolter. Fast is better than free: Revisiting adversarial training. *arXiv:2001.03994*, 2020.

[76] Dongxian Wu et al. Skip connections matter: On the transferability of adversarial examples generated with resnets. *arXiv:2002.05990*, 2020.

[77] C. Xie et al. Mitigating adversarial effects through randomization. In *Proc. of ICLR*, 2018.

[78] Cihang Xie et al. Improving transferability of adversarial examples with input diversity. In *Proc. of CVPR*, 2019.

[79] Ziang Yan, Yiwen Guo, Jian Liang, and Changshui Zhang. Policy-driven attack: Learning to query for hard-label black-box adversarial examples. In *Proc. of ICLR*, 2021.

[80] Zhi Yang, Christo Wilson, Xiao Wang, Tingting Gao, Ben Y. Zhao, and Yafei Dai. Uncovering social network sybils in the wild. *ACM TKDD*, 8(1), 2014.

[81] Yuanshun Yao, Zhujun Xiao, Bolun Wang, Bimal Viswanath, Haitao Zheng, and Ben Y. Zhao. Complexity vs. performance: Empirical analysis of machine learning as a service. In *Proc. of IMC*, Nov. 2017.

[82] Valentina Zantedeschi, Maria-Irina Nicolae, and Ambrish Rawat. Efficient defenses against adversarial attacks. In *Proc. of AISec*, 2017.

[83] Stephan Zheng et al. Improving the robustness of deep neural networks via stability training. In *Proc. of CVPR*, 2016.

[84] Feng Zhou et al. Approximate object location and spam filtering on peer-to-peer systems. In *Proc. of ACM Middleware*, 2003.

Appendix

A Defense against Substitute Model Attacks

Blacklight is designed to detect query based black-box attacks. It cannot defend against attacks transferred from a substitute model. As we discussed in §2, substitute model attacks can be effectively stalled by an existing defense called ensemble adversarial training (EAT) [68]. EAT adversarially trains an ensemble of models with different architectures [49], which are shown to be robust against the substitute model attack. Hence, to defend against all types of black-box attacks, the defender can combine Blacklight with EAT to build a hybrid defense system.

We build and evaluate a hybrid Blacklight and EAT defense on the cifar task. Specifically, we build an ensemble model with three different architectures (6-layer CNN, 8-layer CNN, ResNet-20) and adversarially train the network using PGD attacks as suggested by [49]. We use the same Blacklight configuration as before.

We perform both substitute model based attacks and query based black-box attacks against the above ensemble model defended by Blacklight. For the substitute model attack we run the state-of-art attack proposed by Papernot et al [56], and for the query-based attacks we run the same five black-box attacks. The result shows that the hybrid defense works well and the two defenses do not interfere with each other. The substitute model attack achieves 0% success (thanks to EAT), and Blacklight achieves the same accurate attack query detection as reported before. Thus, we conclude that Blacklight, when combined with EAT, can defend against today's black-box attacks.

Attack	Detection coverage	Avg queries to detect	Attack success w. mitigation	Attack success w/o mitigation
NES - QL	1.8% / 0.8%	52 / 112	97% / 97%	97%
NES - LO	1.3% / 0.9%	52 / 111	85% / 85%	85%
Boundary	1.0% / 0.8%	54 / 115	86% / 86%	86%
ECO	1.8% / 0.9%	53 / 112	88% / 88%	88%
HSJA	1.7% / 0.9%	52 / 111	100% / 100%	100%
QEBA	1.6% / 0.9%	52 / 111	100% / 100%	100%
SurFree	1.9% / 0.9%	52 / 111	100% / 100%	100%
Policy-Driven	2.1% / 0.9%	53 / 111	98% / 98%	98%

Table 8: *Detection performance of Stateful Detection [17] and PRADA [34] when attackers change their accounts after detected and disabled on* `CIFAR10`*, in terms of attack detection and mitigation. The result is presented as s"Stateful Detection / PRADA".*

B Additional Results for §4

Table 8 lists the detection performance of SD and PRADA, when attackers switch to a new account to continue the attack after the current account gets banned.

C Experimental Configurations

Table 9 summarizes the four image classification tasks used in our experiments. More details on associated models, attack configurations and Blacklight's configurations can be found in our extended version [42].

D Additional Results for §8

Boundary attacks with 1 million queries. Table 10 shows that blacklight still has 100% attack detect rates for boundary attacks with 1 million query limits. Furthermore, we find that the detection coverages are even higher for attacks with 1 million query limits than those with 100K query limits. This validates our hypothesis that Blacklight detects boundary attacks at later stage because boundary attack advances slower in converging to the successful adversarial examples. Finally, boundary attacks still have 0% attack success rate with Blacklight mitigation even with 1 million queries.

Blacklight's performance on universal patch attack. Table 11 lists the detailed results for Blacklight's detection and mitigation results on Sparse-RS universal patch attack.

Impacts for Blacklight parameter configuration. We show the experimental results for the impact of Blacklight parameters (Quantization step (**q**), # of hashes per fingerprint (**S**), Sliding window size (**w**), and Sliding step (**p**)) by plotting the Detection Coverage (%) and False Positive Rate (%) with different parameter settings in Figure 10.

E Additional Results for §9

Evasion via Image Transformations. We report the details for our experiments against Image Transformations here. After applying these transformations to attack queries, we report the attack success rate (without the Blacklight defense) and Blacklight's attack detection rate, on the `CIFAR10` task.

Task	Dataset	# Classes	Training data size	Test data size	Input size	Model architecture	Model accuracy
Digit Recognition (`MNIST`)	MNIST	10	60,000	10,000	(28, 28, 1)	6 Conv + 3 Dense	99.36%
Traffic Sign Recognition (`GTSRB`)	GTSRB	43	39,209	12,630	(48, 48, 3)	6 Conv + 3 Dense	97.59%
Object Recognition - Small (`CIFAR10`)	CIFAR-10	10	50,000	10,000	(32, 32, 3)	ResNet20	91.48%
Object Recognition - Large (`ImageNet`)	ImageNet	1000	1,281,167	50,000	(224, 224, 3)	ResNet152	73.05%

Table 9: *Overview of image classification tasks with their associated datasets and models.*

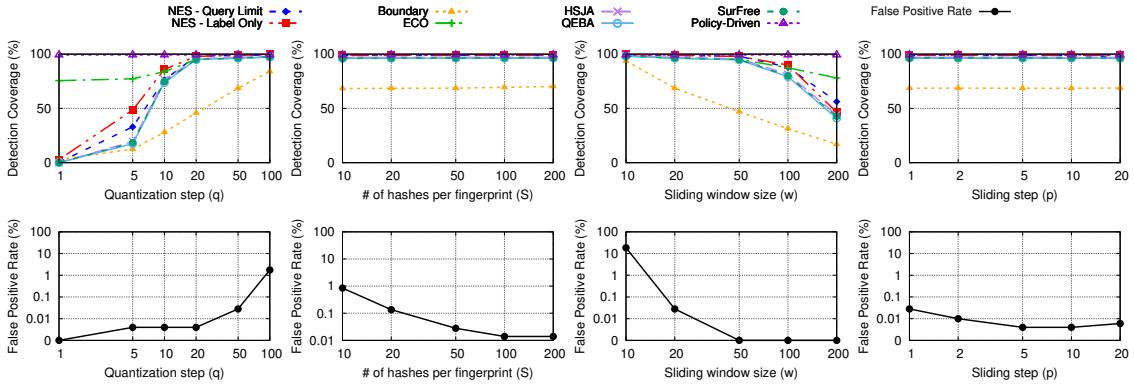

Figure 10: *Detection Coverage (%) and False Positive Rate (%) with different settings on Blacklight parameters: Quantization step (***q***), # of hashes per fingerprint (***S***), Sliding window size (***w***), and Sliding step (***p***).*

Task	w. Detection			w. Mitigation	w/o Blacklight	
	Attack detect %	Detection coverage	Avg queries to detect	Attack success	Attack success	Avg # attack queries
`MNIST`	100%	76.3%	16	0%	26%	892350
`GTSRB`	100%	71.2%	19	0%	40%	902931
`CIFAR10`	100%	69.7%	27	0%	96%	829124
`ImageNet`	100%	97.2%	39	0%	79%	738452

Table 10: *Blacklight's detection and mitigation results on Boundary attack. We stop the boundary attack if it is no successful after 1 million attack queries.*

Task	w. Detection			w. Mitigation	w/o Blacklight	
	Attack detect %	Detection coverage	Avg queries to detect	Attack success	Attack success	Avg # attack queries
`MNIST`	100%	98.4%	8	0%	32.9%	88021
`GTSRB`	100%	98.9%	14	0%	10.8%	98386
`CIFAR10`	100%	97.6%	12	0%	54.7%	87201
`ImageNet`	100%	98.7%	9	0%	27.7%	92039

Table 11: *Blacklight's detection and mitigation results on Sparse-RS universal patch attack.*

Attack \ Transformation		Gaussian Noise w. Different STD				Image Augmentation			
		0.0001	0.0005	0.005	0.05	Shift	Rotate	Zoom	Comb.
NES - QL	**ASR**	85%	80%	15%	0%	100%	75%	80%	60%
	ADR	100%	100%	100%	N/A	100%	100%	100%	100%
NES - LO	**ASR**	25%	20%	15%	0%	100%	45%	70%	20%
	ADR	100%	100%	100%	N/A	100%	100%	100%	100%
Boundary	**ASR**	90%	90%	85%	0%	90%	90%	90%	90%
	ADR	100%	100%	100%	N/A	100%	100%	100%	100%
ECO	**ASR**	85%	0%	0%	0%	0%	0%	0%	0%
	ADR	100%	N/A	N/A	N/A	N/A	N/A	N/A	N/A
HSJA	**ASR**	95%	20%	5%	0%	0%	5%	10%	15%
	ADR	100%	100%	100%	N/A	N/A	100%	100%	100%

Table 12: *Attack success rate (ASR) w/o Blacklight mitigation and Blacklight attack detection rate (ADR) of successful attacks as attackers add different image transformations. Column 3-6 report the results for adding Gaussian Noise with different standard deviation (STD) and Column 7-10 report the results for applying different image transformations to each attack queries.*

Like before, we report attack detection rate only *successful attacks*. For each setting, we run 20 attack instances.

For Gaussian noise based transformations, we vary the standard deviation (STD) of noise from 0.0001 to 0.05 (with all query inputs normalized to [0,1]). Results in Table 12 show that as noise levels increase, attack success rates drop quickly. But at all noise levels tested, Blacklight is able to detect all successful attacks. Intuitively, sufficiently high noise will disrupt classification of both benign and attack queries, thus degrading the attack success rate. We see that Blacklight is generally more robust than the attack's iterative optimization process – Blacklight continues to detect attacks at noise levels where the noise has long since disrupted the attack.

For image augmentation, we test 4 cases where the attacker shifts each input horizontally/vertically by up to 10%, rotate by up to 10°, zoom in by up to 10%, and a combination of all three. Table 12 shows that while different attacks react differently to image augmentation techniques (some still produce successful attacks while others fail completely), Blacklight is able to detect all successful attack sequences under different transformations.

Increasing Perturbation Budget. To provide a comprehensive evaluation on the impact of increasing perturbation budget on Blacklight, we run experiments on all tasks for the two fastest converging attacks (ECO and HSJA) with larger perturbation budgets. Table 13 shows that Blacklight achieves 100% on all tasks for ECO attacks even with perturbation budget up to 0.2. For HSJA attack, Blacklight can de-

Task	ECO				HSJA			
	0.05	0.1	0.15	0.2	0.05	0.1	0.15	0.2
MNIST	100%	100%	100%	100%	100%	100%	75%	40%
GTSRB	100%	100%	100%	100%	100%	100%	70%	50%
CIFAR10	100%	100%	100%	100%	100%	100%	80%	40%
ImageNet	100%	100%	100%	100%	100%	100%	90%	85%

Table 13: *Blacklight detection rate for attacks using larger perturbation budgets for all tasks. We use* $\mathbf{T} = 15$ *with a small increase in false positives (0.74%).*

Attack Type \ N	500	100	50	10
NES - Query Limit	100%	100%	100%	95%
NES - Label Only	100%	100%	100%	31%
Boundary	100%	90%	89%	48%
ECO	100%	100%	100%	100%
HSJA	100%	100%	100%	91%
CW	Average N = 6.33, Detection rate = 100%			
PGD	Average N = 3.13, Detection rate = 81%			

Table 14: *Blacklight's performance against near-optimal "query-efficient" and "perfect-gradient" black-box attacks.*

Attack Type	Average Reset Cycles Needed	Average Total Queries
NES-QL	11471	12695
NES-LO	65837	67099
Boundary	2285	6160
ECO	16590	16591
HSJA	1092	1121

Table 15: *Average reset cycles needed for a successful Pause and Resume attack on* CIFAR10. *The fastest attack (HSJA) can succeed in roughly 3 years.*

tect 100% of attacks on all tasks when the *normalized* L_2 perturbation budgets are no more than 0.1. When the *normalized* L_2 perturbation budgets get larger, Blacklight's detection rate drops gradually. However, we believe this is reasonable since the *normalized* L_2 budget is too large that even exceeds the *normalized* L_2 distances between some benign images. Detailed analysis is listed in our extended version [42].

Pause and Resume Attacks. Table 15 lists the average number of reset cycles needed for different attacks. We also include average total number of queries needed for attacks as reference.

Guided Transformations knowing $(\mathbf{q}, \mathbf{p}, \mathbf{w})$. Algorithm 1 lists the algorithm used by the attacker to generate queries.

Optimal Black-Box Attacks. We provide more details on the optimal black-box attacks. First, to simulate a near-optimal query-efficient attack, we evenly downsample attack query sequences from 5 attacks to generate attack sequences that are a tiny fraction of current sequences. We then test Blacklight's detection performance on these subsampled attack sequences. Table 14 shows that even when attacks are able to complete in 500, 100, or 50 queries, Blacklight still detects them near perfectly (100% detection for 4 attacks and 89% for Boundary attack). Even when these attacks complete within 10 queries, Blacklight is still highly successful

Algorithm 1 Algorithm for Guided Transformation Attacks when Attacker Knows $(\mathbf{q}, \mathbf{p}, \mathbf{w})$

Parameter: Sliding window size $\mathbf{w}$, quantization step $\mathbf{q}$
Input: Attack query x
Output: Modified attack query x

procedure INITIZATION($\mathbf{w}$, $\mathbf{q}$)
    # Save all combinations for pixel modification in a queue.
    PermList $\leftarrow []$
    for $i = 1$ **to** $\mathbf{w}$ **do**
        # compute all $C^i_{\mathbf{w}}$ combinations for selecting i pixels from $\mathbf{w}$ pixels.
        pixelCombination = Combination(i, w)
        # For each pixel selected there are 2 choices for combinations $(+\mathbf{q}/-\mathbf{q})$, which generates $2^i \times C^i_{\mathbf{w}}$ choices in total.
        allPixelCombination = Update2ChoicesPerPixel (pixelCombination)
        PermList.append(allPixelCombination)
    end for
    return PermList
end procedure
procedure GUIDEDTRANSFORMATION(x)
    # we pop the first element from the queue, which is the modification choice with smallest # of pixel changes in the remaining choices.
    CurrentPermutation = PermList.pop()
    Apply the modification for CurrentPermutation to every $\mathbf{w}$ pixels of x.
    return x
end procedure

at detecting NES-QL, ECO and HSJA.

We note that NES-LO and Boundary attacks have much lower detection rates than other attacks when only choosing 10 queries from attack sequences. This is because both NES-LO and Boundary attacks are both boundary attacks that jump back and forth between two images (original and target image). Random subsets of 10 out of thousands of queries are more likely to be variants of the source or target that are sufficiently different from each other as to avoid detection.

Second, for "perfect-gradient" black-box algorithm, each iteration of the gradient calculation for an analogous white-box attack would translate to a single query over the network by the black-box attacker. This idealized black-box attack uses CW [12] and PGD [49] to generate attack sequences against our CIFAR10 model. On average, CW and PGD converge after only 6.3 and 3.1 queries. Against simulated black-box attacks using these attack queries, Blacklight detects 100% of attacks driven by CW, and 81% of PGD-driven attacks.