Stateful Defenses for Machine Learning Models Are Not Yet Secure Against Black-box Attacks

Ryan Feng[†]
rtfeng@umich.edu
University of Michigan
Ann Arbor, Michigan, USA

Ashish Hooda[†]
ahooda@wisc.edu
University of Wisconsin-Madison
Madison, Wisconsin, USA

Neal Mangaokar[†]
nealmgkr@umich.edu
University of Michigan
Ann Arbor, Michigan, USA

Kassem Fawaz
kfawaz@wisc.edu
University of Wisconsin-Madison
Madison, Wisconsin, USA

Somesh Jha
jha@cs.wisc.edu
University of Wisconsin-Madison
Madison, Wisconsin, USA

Atul Prakash
aprakash@umich.edu
University of Michigan
Ann Arbor, Michigan, USA

ABSTRACT

Recent work has proposed *stateful defense models (SDMs)* as a compelling strategy to defend against a black-box attacker who only has query access to the model, as is common for online machine learning platforms. Such stateful defenses aim to defend against black-box attacks by tracking the query history and detecting and rejecting queries that are "similar" and thus preventing black-box attacks from finding useful gradients and making progress towards finding adversarial attacks within a reasonable query budget. Recent SDMs (e.g., Blacklight and PIHA) have shown remarkable success in defending against state-of-the-art black-box attacks. In this paper, we show that SDMs are highly vulnerable to a new class of adaptive black-box attacks. We propose a novel adaptive black-box attack strategy called Oracle-guided Adaptive Rejection Sampling (OARS) that involves two stages: (1) use initial query patterns to infer key properties about an SDM's defense; and, (2) leverage those extracted properties to design subsequent query patterns to evade the SDM's defense while making progress towards finding adversarial inputs. OARS is broadly applicable as an enhancement to existing black-box attacks – we show how to apply the strategy to enhance six common black-box attacks to be more effective against current class of SDMs. For example, OARS-enhanced versions of black-box attacks improved attack success rate against recent stateful defenses from almost 0% to to almost 100% for multiple datasets within reasonable query budgets.

CCS CONCEPTS

• **Computing methodologies** → **Machine learning**; • **Security and privacy**;

KEYWORDS

Machine Learning, Adversarial Examples, Security, Black-box Attacks, Stateful Defenses

ACM Reference Format:
Ryan Feng, **Ashish Hooda**, **Neal Mangaokar**, Kassem Fawaz, Somesh Jha, and Atul Prakash. 2023. Stateful Defenses for Machine Learning Models Are Not Yet Secure Against Black-box Attacks. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security (CCS '23), November 26–30, 2023, Copenhagen, Denmark.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3576915.3623116

1 INTRODUCTION

Machine learning (ML) models are vulnerable to adversarial examples, imperceptibly modified inputs that a model misclassifies. Adversarial examples pose a significant threat to the deployment of ML models for applications such as deepfake detection [24, 37], autonomous driving [20, 29], medical image classification [22, 43], or identity verification [40]. Unfortunately, crafting defenses against white-box attackers who have full model access has proven difficult, particularly with the advent of adaptive attack strategies. There remains a large gap between natural and adversarial performance [3, 7, 13, 18, 21, 31, 41].

In recent years, there has been an increasing focus on a more restricted, but realistic black-box attack threat model, where an adversary only has query access to the model, e.g., via an API exposed by Machine Learning as a Service (MLaaS) platforms [1, 12, 38]. The attacker can query the model for labels or label probability scores, but has no further access to the model or its training data. Several successful black-box attack methods have been proposed that use gradient or boundary estimation techniques to construct adversarial examples [6, 8, 19, 25, 30, 33, 34, 44, 45]. However, such techniques typically require querying multiple nearby inputs to approximate the local loss landscape.

This observation has led to a new line of defense work we refer to as *Stateful Defense Models (SDMs)* [9, 11, 26, 29], which target black-box query-based attacks. Observing that such attacks sample multiple nearby points, SDMs use an internal state to track past queries. SDMs then monitor future queries and perform some restrictive action against the attacker when receiving queries that are too similar, where similarity is defined through some defense-chosen measure. Such an event is also referred to as a *collision*. Defensive actions can then include banning the user's account or rejecting

† Denotes equal contribution.

CCS '23, November 26–30, 2023, Copenhagen, Denmark

ACM ISBN 979-8-4007-0050-7/23/11…$15.00
https://doi.org/10.1145/3576915.3623116

queries. Blacklight [29] is a state-of-the-art SDM, which uses probabilistic content fingerprints to reject highly similar queries, thereby thwarting black-box attacks.

Since SDMs directly target the black-box attacks' fundamental reliance on issuing similar queries, attacking SDMs remains an open and challenging problem. Most contemporary black-box query-based attacks involve some combination of gradient estimation from averaging samples drawn from a distribution and walking along the decision boundary in small steps, both of which can easily result in account bans or query rejections (Section 3.1). The main method for adapting black-box attacks is query-blinding [9], which applies simple transformations on input queries. It aims to make small changes to avoid detection without disrupting the attack optimization process. Blacklight [29], in particular, have made remarkable progress, with a perfect 0% attack success rate against a wide range of attacks, even with query-blinding (an observation we confirm in Section 4).

However, a related question naturally arises: Similarly to the white-box setting, where robustness was overstated due to a lack of adaptive attack evaluation [3], can SDMs be broken by stronger adaptive black-box attacks that attempt to evade query collisions?

In this paper, we find that SDMs are highly vulnerable to a new class of adaptive black-box attacks. *The key insight underlying these attacks is that SDMs leak information about their similarity-detection procedure.* We use this information to adapt and enhance black-box attacks to be more effective against these SDMs. Our novel adaptive black-box attack strategy called Oracle-guided Adaptive Rejection Sampling (OARS) involves two stages: (1) use initial query patterns to extract information about the SDM's similarity-detection procedure; and (2) leverage this knowledge to design subsequent query patterns that evade the SDM's similarity check while making progress towards finding adversarial inputs (Section 3).

Using OARS to work with a black-box attack is a non-trivial design problem – multiple elements in a typical black-box attack must be modified to achieve the following tasks, while *avoiding collisions*: (1) sampling to estimate gradients; (2) choosing an appropriate step size; and (3) modifying the technique for finding the decision boundary in hard-label black-box settings. We show how OARS applies such modifications in a principled way. First, we query the model to fine-tune a proposal probability distribution of the relevant parameters to each task. Second, we repeatedly sample the fine-tuned distribution to generate examples and use the defense as an oracle to reject or accept them. The accepted examples evade collisions and lead to more successful attacks (Section 3.1).

We demonstrate how OARS can broadly enhance a wide range of black-box attacks to make them more potent against SDMs. In particular, we apply OARS to six commonly used black-box query-based attacks (NES [25], HSJA [8], QEBA [30], SurFree [33], Square [2], Boundary [6]) (Section 3.2). Through comprehensive empirical evaluation, we find that OARS-enhanced versions of these black-box attacks significantly outperform both the standard versions and query-blinding on four contemporary stateful defenses (the original stateful detection defense [9], Blacklight [29], PIHA [11], and IIoT-SDA [17]). For the best OARS-enhanced black-box attacks, the attack success rate increased from close to 0% (for the best defense) to almost 100% for each of the four stateful defenses on multiple datasets with reasonable query budgets for the attacks to be practical (Section 4.2). Finally, we discuss potential directions for improving stateful defenses to counteract our new attacks. We tested several variations of existing SDMs to counteract our attacks but found that OARS-enhanced black-box attacks maintained a high attack success rate (Section 4.3).

In summary, this paper demonstrates that recent SDMs, thought to be strong defenses against black-box attacks, are actually highly vulnerable. We propose a technique called Oracle-guided Adaptive Rejection Sampling (OARS) that helps make multiple black-box attack algorithms much more potent against these SDMs. These OARS-enhanced black-box attack methods thus provide a new benchmark to test any future proposed stateful defenses against black-box attacks.

2 BACKGROUND AND RELATED WORK

In this section, we describe our notation and terminology, threat model, black-box attacks, stateful defense models (SDMs), and the difficulties of attacking stateful defense models.

2.1 Notation and Terminology

First, we introduce our common notation and terminology to describe the SDMs and the related attacks.

2.1.1 ***General Notation.*** Let $\mathcal{D}$ be the distribution of input space $\mathcal{X} \times \mathcal{Y}$, where $\mathcal{X} \in \mathbb{R}^d$ is the space of $d-$dimensional samples, and $\mathcal{Y}$ is the class label space. Let $F : \mathcal{X} \to [0,1]^{|\mathcal{Y}|}$ be the "soft-label" DNN classifier trained using loss function L that outputs probabilities over the classes, i.e., $F(x)_i$ is the probability that $x \in \mathcal{X}$ belongs to the i^{th} class. The "hard-label" classifier can then be given by $f(x) = \arg\max_i \ F(x)_i$. Given a victim sample $x_{vic} \in \mathcal{X}$ with true label $y \in \mathcal{Y}$, and a perturbation budget ϵ, an adversarial example is any sample x_{adv} such that $f(x_{adv}) \neq y$ and $||x_{adv} - x_{vic}|| \leq \epsilon$ for an appropriate choice of norm. We describe an identity matrix in $\mathbb{R}^d$ by I_d.

For iterative black-box attacks, let x_t represent the sample at the current t^{th} iteration of the attack.

2.1.2 ***SDM Terminology.*** To establish a standard setup for SDMs, we introduce the six components that make up a typical SDM. We then describe the SDMs we evaluate in terms of these named components in Section 2.4.

Classifier. The classifier F is the underlying model to be protected by the SDM.

Feature Extractor. The feature extractor is a function $h(\cdot)$ that extracts a representation of incoming queries to determine query similarity. The similarity of two samples x_1 and x_2 is the distance between $h(x_1)$ and $h(x_2)$.

Query Store. The query store is a finite-capacity stateful buffer Q that stores the history of past queries.

Similarity Procedure. The similarity procedure s uses $h(\cdot)$ to compute the distance of a given query to the most similar element(s) in the store.

Action Function. The action function `action` identifies collisions and takes the necessary steps to thwart the attack. Informally,

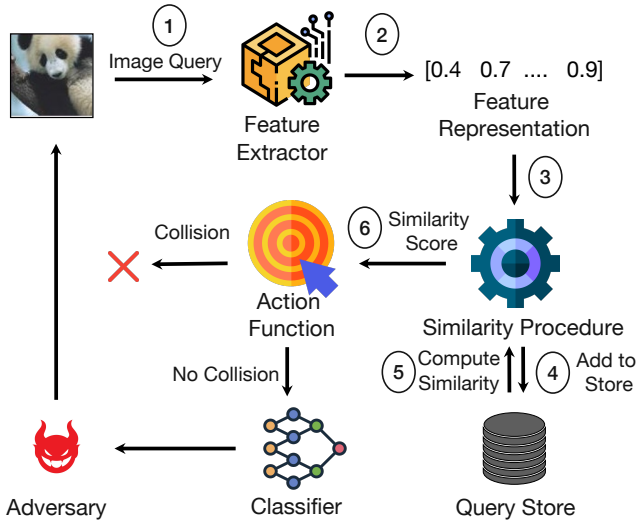


Figure 1: A general stateful defense pipeline. Given an input query, the input's features are compared for similarity with features of queries stored in the query store, resulting in a similarity score. It also adds the query to the query store. The action function then decides whether the similarity score indicates a collision. In case of no collision, the model is queried, and the model output is returned; else, an action is taken, such as rejecting the query.

a collision occurs when the similarity procedure determines the incoming query to be "too similar" to a previous query. We consider SDMs with two different assumptions about user accounts (see Section 2.4). Depending on the underlying assumptions of the attackers, SDMs either act by banning the accounts [9] or rejecting queries [11, 17, 29] to limit the attacker.

Figure 1 shows a conceptual pipeline of SDMs in terms of the above elements. Given an input query (for example, an image to be classified), the input sample passes through a feature extractor. The output feature representation is then passed to the similarity procedure, which compares this representation with those in the query store and outputs a similarity score. It also adds the query to the query store. The action function then decides whether the similarity score indicates a collision or not (a collision implies the query is similar to a stored query). If there is no collision, the model is queried, and the model output is returned. If there is a collision, typical action is to reject the query or ban a user's account.

2.2 Threat Model

We consider a black-box threat model where a queryable Machine-Learning-as-a-Service (MLaaS) platform hosts a classifier. Examples of such platforms include Clarifai [12], Amazon Rekognition [1], and Automatic License Plate Recognition systems [38]. Users can submit queries by registering an account with the service and interacting with its public API. We focus on query-based attacks, where an attacker can only query the model for outputs, which can either be just the final label (hard-label) or include class probabilities (soft-label or score-based). We evaluate our SDMs against both score-based and hard-label query-based attacks [2, 6, 8, 25, 30, 33]. *The proposed OARS attacks are model agnostic - we do not utilize any knowledge about the underlying model specifics, whether an SDM is deployed at all, or which specific SDM method if deployed, is being used.*

If an SDM is deployed, we do not disable or pause the SDM for any stage of the attack in any of our experiments. This means that SDMs run unmodified for the entire duration of the attack and are free to take action by rejecting queries or banning accounts during all stages of our attacks, matching a practical deployment where the SDM will always be active. Our key insight is that OARS can leverage these rejections or bans to adapt the attacks.

2.3 Black-box Attacks

We analyze a set of black-box attacks this paper: NES [25], HSJA [8], QEBA [30], Boundary [6], Square [2], and SurFree [33], which we briefly describe below. Following prior work [29], we use ℓ_∞ versions of NES [25] and Square [2] and ℓ_2 versions of the remaining attacks.

NES (Score-based) [25]. NES is a score-based attack that aims to construct adversarial examples by estimating the loss gradient. NES uses finite differences over samples from a Gaussian distribution to estimate the gradient of the loss. It then performs projected gradient descent with the estimated gradient.

Square (Score-based) [2]. Square attack is a score-based attack that applies ℓ_p-bounded random perturbations to pixels within the squares inside the input sample. It chooses progressively smaller squares to ensure attack success.

HSJA (Hard-label) [8]. HopSkipJumpAttack (HSJA) is a hard-label attack that repeatedly (1) locates the model decision boundary via binary search, (2) estimates the gradient around the boundary via estimated local loss differences, and (3) identifies the optimal step size via geometric progression before performing gradient descent.

QEBA (Hard-label) [30]. QEBA is a hard-label attack based on HopSkipJumpAttack that samples noise from a lower-dimensional subspace to estimate the gradient around the boundary.

SurFree (Hard-label) [33]. SurFree is a hard-label attack that avoids gradient estimation. Guided by the geometric properties of the decision boundary, it performs a random search of directions starting from a given sample to identify the direction closest to the decision boundary.

Boundary (Hard-label) [6]. Boundary is a hard-label attack that starts from a randomly initialized adversarial example and performs a "boundary walk" to reduce the perturbation size.

2.4 Stateful Defense Models

The key intuition behind stateful defense models is that query-based black-box attacks need to query "similar" samples. Thus, rejecting any query that is "too similar" to a previously queried sample, as described in Section 2.1.2, can help prevent these attacks from estimating accurate gradients. For example, NES [25] samples multiple Gaussian perturbed versions of a given sample x_t to estimate the

Table 1: Existing defenses (OSD [9], Blacklight [29], PIHA [11], and IIoT-SDA [17]) summarized in terms of their choices for each component.

Defense	Query Store	Feature Extractor	Similarity Procedure	Action
OSD [9]	Per Account	Neural Encoder	kNN over ℓ_2 Distance ($k = 50$)	Ban Account
Blacklight [29]	Global	Pixel-SHA	kNN over Hamming Distance ($k = 1$)	Reject Query
PIHA [11]	Global	PIHA's Percept. Hash	kNN over Hamming Distance ($k = 1$)	Reject Query
IIoT-SDA [17]	Per Account	Neural Encoder	kNN over ℓ_2 Distance ($k = 11$)	Reject Query

loss gradient, with such queries likely being similar to the original query x_t.

We now describe four existing SDMs (OSD [9], Blacklight [29], PIHA [11], and IIoT-SDA [17]) using the terminology in Section 2.1.2. Table 1 compares these SDMs.

OSD [9]. Originally proposed by Chen et al. [9], OSD employs a neural similarity encoder as the feature extractor h. The encoder outputs a $d-$dimensional embedding vector and is trained using contrastive loss [5] to learn perceptual similarity. To detect whether incoming query x induces a collision, OSD uses a per-account query store and similarity procedure that computes the average ℓ_2 distance between $h(x)$ and its $k-$nearest neighbors in that account's query store. Finally, OSD's action function bans the user's account if a collision is detected. As Sybil accounts [15, 46] successfully render the cost of multiple accounts minimal, OSD can be trivially broken. Blacklight [29] and PIHA [11] address this limitation.

Blacklight [29]. Blacklight [29] is an SDM that differs from OSD [9] in two ways. First, it changes the feature extractor h to a new probabilistic hash function, Pixel-SHA. Pixel-SHA quantizes pixel values of the input sample, hashes multiple segments of pixels at a given stride length, and concatenates the top 50 hashes to form the final output. Since the output of h is a hash, Blacklight employs an approximate nearest-neighbor search over normalized Hamming distances (which range between 0 and 1) as its similarity procedure, which can be computed in $O(1)$ time.

Second, with the assumption that Sybil accounts [15, 46] can be easily created, Blacklight [29] employs a global hash-table as its query store. Since the store is global, collisions are detected regardless of the number of accounts the attacker creates. As a result of this change, Blacklight's action function chooses to reject queries, i.e., deny classification service when a collision is detected, instead of banning accounts.

Blacklight assumes that the global store is large but finite and needs to be reset infrequently once a day to handle the query workload. With a reset of once per day, Blacklight authors report that it would take an attacker approximately 3 years to complete the fastest successful attack if they waited for the store to reset before proceeding, which is a significant slowdown.

PIHA [11]. PIHA [11] is an image-specific SDM similar to Blacklight [29] in that it uses a hash function to detect similarities and uses a global store to detect collisions. PIHA, however, proposes a novel perceptual hashing algorithm to act as a feature extractor. This algorithm applies a low-pass filter, converts the image into the HSV or YC_BC_R color spaces, and uses the Local Binary Patterns algorithm [35] to compute the hash function on blocks of the input.

IIoT-SDA [17]. IIoT-SDA [17] is a defense targeting malware classification in an industrial IoT setting. IIoT-SDA reshapes executable byte codes into a matrix and trains a neural encoder with a Mahalanobis [32] defined contrastive loss function as its feature extractor h. IIoT-SDA uses a per-account query store and k-nearest neighbors as its similarity procedure, like OSD [9]. Like Blacklight [29], IIoT-SDA's action function is to reject queries. Thus, IIoT-SDA can be viewed as a combination of OSD and Blacklight, adapted to the malware classification domain.

2.5 Query-blinding Attacks

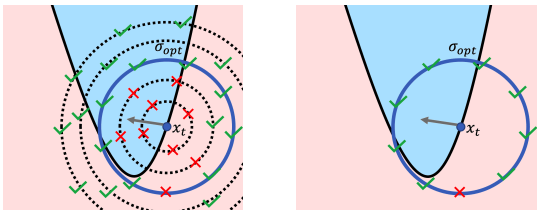
SDMs have shown promise against black-box query-based attacks. For example, Blacklight [29] reports 0% attack success rate for all of the black-box attacks in Section 2.3. We attribute this success to attacks issuing similar queries to the target model. To avoid issuing similar queries, Chen et al. proposed *Query-blinding attacks* [9] that attempt to evade SDM detection by transforming samples before performing queries. The intuition is that it may be possible to transform samples so that the samples are no longer "similar" enough to be detected by an SDM but useful enough to infer gradients and enable the underlying optimization process to succeed.

Formally, query-blinding is defined by two functions: a randomized blinding function $b(x; s)$ that maps the input query x to a modified example x' such that x and x' do not result in a collision, and a revealing function $r(F(x'))$ that estimates $F(x)$ from the blinding function and the classifier outputs ($F(x')$). For the image domain, query-blinding is most commonly implemented by taking image transformations (e.g., rotation, translation, scaling, Gaussian noise) and selecting a random value for each query within a range of values parameterized by s.

Still, Blacklight [29] and recent SDMs [9, 11] claim to defeat query-blinding attacks, suggesting that attacking SDMs remains a difficult and unsolved problem. In particular, query-blinding struggles with balancing the trade-off between attack utility and detection: more destructive transforms increase the odds of evading detection but also hurt the optimization process. We provide further experimental insight into why query-blinding fails in Section 4.

3 DESIGNING ADAPTIVE BLACK-BOX ATTACKS

We propose a novel adaptive black-box attack strategy against SDMs based on the following insight: *the SDM's action module "leaks" information about its similarity procedure and query store.* An attacker can leverage information from the defender's past action behavior to issue queries that evade collisions while performing the attack. More formally, we represent the SDM as an oracle that



(a) ***OARS-NES: Adapting the proposal distribution.*** **For gradient estimation, we adapt a Gaussian proposal distribution to estimate the optimal σ_{opt} that achieves the target collision rate.**

(b) ***OARS-NES: Resampling.*** **For gradient estimation, once we have σ_{opt}, we perform rejection sampling over from $\mathcal{N}(0, \sigma^2_{opt} I_d)$. We resample up to ge_{tries} to get n valid samples and average over the valid samples we obtain.**

Figure 2: Example illustration of the OARS adapt and resample gradient estimation for NES [25]. Red X's denote queries that collide, and green checkmarks denote queries that do not collide.

enables the attacker to perform *rejection sampling* (discussed further in Section 3.1) to help future queries avoid collisions. This insight guides our proposed Oracle-guided Adaptive Rejection Sampling (OARS) strategy, which introduces a two-pronged *adapt* and *resample* strategy to evade collisions.

We first describe three key elements of black-box attacks and how the attacker adapts each element using rejection sampling (Section 3.1). Then, we apply our general adaptive strategy to enhance six popular black-box attacks (NES [25], HSJA [8], QEBA [30], SurFree [33], Square [2], and Boundary [6]) (Section 3.2).

3.1 Modifying Common Attack Elements with Oracle-Guided Adaptive Rejection Sampling

We begin with the following observation: Typical query-based attacks in the black-box settings involve three attack elements: estimating a gradient, taking a step, and locating the boundary.

- **Estimating a gradient:** Given a current attack point x_t, gradient estimation typically adds Gaussian noise scaled by a standard deviation of σ and compares the differences in model outputs.
- **Taking a step:** After estimating the gradient, attackers update x_t by taking a step in that direction. Black-box attacks typically anneal their step size to small values for improving convergence.
- **Locating the boundary:** After taking a step, many black-box attacks update the example by interpolating the line between the post-step attack point x_t and the original sample x_{vic}. This interpolation moves the x_t towards the boundary to select a better starting point for the next iteration. Typically, this takes the form of a binary search between x_t and x_{vic}.

An effective adaptive attack against SDMs must revisit these elements to issue queries that evade collisions while converging to a successful adversarial example. Towards that end, we propose a general strategy, Oracle-guided Adaptive Rejection Sampling (OARS),

Algorithm 1 ADAPT_PROPOSAL: Fine-tune a parametric proposal distribution $\mathcal{N}(0, \sigma^2 I_d)$ via Oracle-guided binary search for the gradient estimation attack element.

Input: Proposal Distribution $\mathcal{N}(0, \sigma^2 I_d)$, Oracle Access to SDM, bounds σ_{lo} and σ_{hi}, a number of steps $stps$, a number of samples sam and max collision rate cr

Output: Fine-tuned σ_{opt} for circumventing the SDM with a collision rate of cr

for $stps$ steps of binary search **do**
  $\sigma_{mid} \leftarrow (\sigma_{lo} + \sigma_{hi}) / 2$
  Generate sam samples from $\mathcal{N}(0, \sigma^2_{mid} I_d)$
  Query the SDM sam times
  collision rate $\leftarrow$ ratio of rejected queries
  if collision rate $> cr$ **then**
    Select the upper half range, $\sigma_{lo} \leftarrow \sigma_{mid}$
  else
    Select the lower half range, $\sigma_{hi} \leftarrow \sigma_{mid}$
  end if
end for
$\sigma_{opt} \leftarrow \sigma_{hi}$
return σ_{opt}

which leverages the defense to adapt attack queries. Recall that SDMs thwart query-based black-box attacks by rejecting "similar" queries. OARS uses the SDM to select the most similar queries that will not collide with a given sample x. This way, it uses rejection sampling to pick queries from a target distribution $p'_x(q)$, which represents the most similar potential queries that won't cause a collision with the sample x. More specifically, rejection sampling selects a tractable *proposal distribution* $p^\theta_x(q)$ as an estimate of $p'_x(q)$, and then samples a query from $p^\theta_x(q)$. If this query collides, it can be discarded, and a new sample be drawn.

However, selecting a good proposal distribution, i.e., selecting a good θ poses a key challenge — sampling from a poorly selected proposal distribution will yield a large number of samples that do not belong to $p'_x(q)$, i.e., submitting these queries causes frequent collisions and exhausts the query budget. To address this challenge, OARS initializes a parametric proposal distribution $p^\theta_x(q)$ and uses the SDM as an oracle to find an estimated θ_{opt} such that $p^{\theta_{opt}}_x(q) \approx p'_x(q)$. We assume that θ is a unimodal and monotonic parameter, i.e., the average collision rate decreases with an increase in θ. Adaptation can be achieved via any suitable optimization algorithm. For example, an adversary can use the binary-search protocol in Algorithm 1 to fine-tune their $p^\theta_x(q)$.

OARS thus includes a two-pronged *adapt and resample* strategy. First, the adversary *adapts* the proposal distribution, and samples from this distribution to construct attack queries. If, at any stage, a query causes a collision, the adversary can simply *resample* a substitute query from the proposal distribution. OARS is a general approach that applies to any attack consisting of the above three attack elements. It does not make any assumptions about the SDM being deployed — the SDM remains in place at all times and OARS must handle any rejections/bans. In the rest of this section, we describe our strategy for choosing the proposal distributions. Then,

Algorithm 2 OARS: Oracle-guided Adaptive Rejection Sampling for the gradient estimation attack element.

Input: Current sample x, Proposal Distribution $\mathcal{N}(0, \sigma^2 I_d)$ around current sample, Oracle Access of SDM, number of samples n, max retries ge_{tries}
Output: Model output for n sampled queries

```
Optimal Parameter σ_opt ← ADAPT_PROPOSAL(N(0, σ² I_d))
while # successful queries < n AND # queries < ge_tries do
    Resample q ← N(0, σ²_opt I_d)
    Query the model using x + q
end while
return successfully sampled queries
```

we describe how three standard attack components can be adapted using OARS to attack SDMs successfully.

3.1.1 ***Strategy for Choosing Proposal Distributions.*** Our rationale for choosing the proposal distributions is to minimize changes to the vanilla attacks as much as possible. We note that in vanilla versions of the attacks, each attack element either samples (1) from a fixed distribution (e.g., Gaussian with fixed variance) or (2) based on a fixed parameter (e.g., fixed step-size). As such, two cases arise:

- The first case is when performing stochastic operations, such as sampling from a Gaussian distribution or a uniform distribution. In such a case, OARS adopts the same distribution with an important change: adapt the parameters of that distribution. For example, since base/vanilla NES samples from a Gaussian distribution for gradient estimation, OARS-NES assumes a Gaussian with tunable variance as the proposal distribution, adapts the variance, and then samples from it. Another example is the HSJA attack which uniformly samples from a sphere of a fixed radius. OARS-HSJA uses a uniform distribution of the sphere radius as the proposal distribution.
- The second case is when queries are based on a fixed parameter. Here, we construct the proposal distribution as uniformly distributed in the subspace of the queries resulting from the operation with the appropriate tunable parameter. For example, when taking a step, vanilla NES takes a fixed-sized step in the direction of the sign of the gradient. The proposal distribution in OARS-NES samples queries uniformly over all possible step directions, captured by the Rademacher distribution parameterized by step size. After that, the samples are scaled by the step size (the tunable parameter which is adapted during the adapt phase).

3.1.2 ***Estimating a gradient.*** Gradient estimation typically samples points around the current sample (x_t) by adding a noise term from a distribution such as an isotropic Gaussian $\mathcal{N}(0, \sigma^2 I_d)$ (parameterized by σ) or uniform distribution on a sphere. A small value of σ results in sampling points within a small neighborhood around x_t. In that case, the attack will converge faster at the expense of query collisions. For example, during gradient estimation, NES samples n close-by variants of a given starting point x_t. It then performs localized differences to estimate the gradient.

OARS Variant. We apply the OARS-based adapt and resample strategy to construct a proposal distribution for sampling gradient estimation queries that avoid a collision. First, we adapt the sampling distribution used by the underlying attack (e.g., Gaussian with zero mean and parameterized by variance, $\mathcal{N}(0, \sigma^2 I_d)$ for NES). However, unlike current attacks that use a fixed σ, we *adapt* the proposal distribution by performing an oracle-guided fine-tuning of σ using binary search. The entire process can be seen in Algorithm 1 and illustrated in Figure 2a for NES. Specifically, the binary search samples *sam* queries at each point evaluated between a given σ_{lo} and σ_{hi}. The search is run for $stps$ steps, converging at the smallest distance that results in less than the tolerable collision rate cr. Finally, one can sample the gradient estimation samples from $\mathcal{N}(0, \sigma_{opt}^2 I_d)$ where ${\sigma_{opt}}^2$ is the fine-tuned variance. A similar process can be used if a different distribution is used instead of a Gaussian.

Second, we perform rejection sampling using the fine-tuned proposal distribution to get the model output for n points, which are then used to estimate the gradient by averaging the loss differences at the n samples. To deal with possible collisions, our strategy is to *resample* from the proposal distribution up to a given ge_{tries} times to ensure we get all n valid samples. Algorithm 2 details this process (and Figure 2b also illustrates it for NES). However, even after trying for ge_{tries} times, if we still do not have n valid samples, we move forward - as long as we have 1 valid sample, we average over the valid samples we have. Figure 2a illustrates the adaptive gradient estimation for NES.

3.1.3 ***Taking a Step.*** The step size for this attack element presents a trade-off between collision and convergence. For example, HSJA takes the largest step size possible (such that the resultant point is still adversarial) and decreases this value over time to speed up convergence. However, if the attack queries use a small step, the adversary risks a collision and could fail to make progress toward generating an adversarial example.

OARS Variant. We apply the OARS adapt and resample strategy to construct a proposal distribution to sample points using the smallest step that avoids collisions. First, we assume a proposal distribution that models steps taken by the attack. For instance, a suitable distribution could be (a) a $\lambda-$scaled Rademacher distribution for the NES attack, to represent steps taken towards the sign of the gradient or (b) a uniform distribution on the surface of a sphere of radius λ for the HSJA attack, to represent steps taken in the direction of the gradient. Here, λ is the tunable step size. Then, much like Algorithm 1 adapts a Gaussian for gradient estimation, we *adapt* the proposal distribution by performing an Oracle-guided fine-tuning of λ using binary search resulting in λ_{opt}. Finally, given a gradient direction, one can sample the step from the proposal distribution conditioned on the gradient direction. Second, in the case of a collision, we *resample* the corresponding step up to $steps_{tries}$ times along different gradient directions (similar to the resampling procedure for gradient estimation in Algorithm 2).

3.1.4 ***Locating the Boundary.*** Recall that the attacker locates a decision boundary by interpolating along a line from the current attack sample towards the original input sample. This interpolation is a binary search where the minimum distance between successive

queries decreases exponentially. If this distance is too small at any point during this interpolation, the query may collide and prevent attack progress.

OARS Variant. We define a binary search operation: $\phi(r)$, which proceeds until the distance between successive queries is less than the distance r, which we define as the termination distance. We apply the OARS adapt and resample strategy to construct a proposal distribution for sampling the termination distance for the interpolation binary search. First, we use a uniform proposal distribution parameterized by the upper bound $a : U(0, a)$. This upper bound is initialized by the distance between current sample x_t and victim sample x_{vic}. Then, we *adapt* the proposal distribution by performing an oracle-guided fine-tuning of a using binary search. Second, in case of an unsuccessful binary search, we resample another termination distance and perform the binary search $\phi(r)$ again. In the implementation, we efficiently perform this *resampling* by reusing the unsuccessful binary search and taking the penultimate query (equivalent to a binary search operation with twice the termination distance).

3.1.5 ***Determining SDM Store.*** We observe that the cost of including our OARS strategy depends on the specific SDM being deployed. For query rejection-based defenses, i.e., Blacklight [29], PIHA [11] and IIoT-SDA [17], the additional query overhead from OARS can be run without incurring additional account requirements. However, for account banning-based defenses like OSD [9], the additional overhead may require more accounts than is needed to perform the standard attack across multiple steps (Section 4.2). As such, we devise a test to determine 1) if the defense is banning accounts and 2) if it is using a per-account or global query store. This test has two steps. First, the attack creates an account A and feeds it in an input x. It queries x until it is detected on account A (or up until a certain limit), suggesting that SDMs are likely not being used. Second, the attacker creates account B and queries x again on that account (e.g., a query that was detected using account A). If it is detected, the system must be using a global query store. Otherwise, we assume that it is using a per-account scheme. The cost of this test is the number of queries required for detection on A, the additional query on B, and the additional account.

3.2 Modifying Existing Attacks with OARS

We now describe how to modify six existing black-box attacks to demonstrate how to apply our adaptations to a variety of attacks. Figure 2 shows diagrams visualizing the changes for the gradient estimation stage of NES [25]. Here, we describe the attack components that are modified when integrated with OARS. We also summarize which elements are modified for the specific attacks we analyze in Table 2. Source code for our modified black-box attacks is available at https://github.com/nmangaokar/ccs_23_oars_stateful_attacks.

While we demonstrate specific modifications for six attacks, we note that the OARS approach of adapting proposal distributions and resampling queries generalizes to attacks with the three elements discussed in Section 3.1. The discussions below cover a range of differing attacks and provide blueprints for adapting other attacks. This is done by first identifying the three common elements and applying the corresponding modifications to adapt proposal distributions and resample as applicable.

Table 2: Table summarizing the adaptive modifications to these attacks. Full circle means that we apply OARS's adapt and resample to modify this element. Empty circle means that the attack does not have this element.

Attack	Est. a Gradient	Taking a Step	Locating the Boundary
OARS-NES	●	●	○
OARS-Square	○	●	○
OARS-HSJA	●	●	●
OARS-QEBA	●	●	●
OARS-SurFree	○	●	●
OARS-Boundary	○	●	●

3.2.1 ***OARS-NES (Score-based).*** To create OARS-NES, we use OARS to modify vanilla NES's [25] gradient estimation (estimating a gradient) and projected gradient descent (taking a step) to avoid detection with more dissimilar queries.

Estimating a Gradient. Vanilla NES [25] estimates a gradient using finite differences over n Monte Carlo samples from a Gaussian distribution as shown below, where L is the loss function to be estimated, σ^2 is the variance of the distribution, and $u_i \sim \mathcal{N}(0, I)$:

$$\nabla^{est}_{x_t} L = \frac{1}{\sigma n} \sum_{i=1}^{n} [L(x_t + \sigma u_i,\ y) u_i] \tag{1}$$

Modified: *We use a Gaussian proposal distribution $\mathcal{N}(0, \sigma^2 I_d)$ to sample the gradient estimation queries that avoid collision. Then, we use OARS's adapt & resample to fine-tune σ and generate n queries for gradient estimation. This is similar to the approach described in Section 3.1.2*

Taking a Step. For taking a step, vanilla NES updates with a fixed step size λ with projected gradient descent:

$$x_{t+1} = \text{Proj}_{x_{orig}+\mathcal{S}}(x_t + \lambda \cdot \text{sgn}(\nabla^{est}_{x_t} L)) \tag{2}$$

where $\mathcal{S} = \{\delta : ||\delta|| \leq \epsilon\}$ and Proj is the projection operator.

Modified: *We use a λ-scaled Rademacher distribution where $0 \leq \lambda \leq \epsilon$ to sample the smallest step queries that avoid collision. Then, we use OARS's adapt & resample to fine-tune λ and sample the next step. This is similar to the approach described in Section 3.1.3*

3.2.2 ***OARS-Square (Score-based).*** To create OARS-Square, we use OARS to modify the number of the random square perturbations being sampled at each step. Square attack does not have a gradient estimation step.

Taking a Step. The ℓ_∞ version of vanilla Square [2] samples random squares with perturbations filled to $-\epsilon$ or ϵ and selecting the square that increases the loss function. In particular, it samples random squares of progressively smaller sizes to help the attack converge.

Modified: *We apply OARS's adapt and resample for sampling squares while evading detection by the SDM. For each square size, we first modify the sampling process of the Square attack to select multiple*

squares for each step. Then, we construct a proposal distribution for sampling the minimum number of squares required to avoid a collision. We then apply OARS's adapt mechanism to fine-tune the proposal distribution. Finally, we use OARS's resample mechanism to handle collisions. When the attack proceeds to a smaller square size, we repeat the process.

3.2.3 ***OARS-HSJA (Hard-label).*** To create OARS-HSJA, we use OARS to modify vanilla HSJA's [8] gradient estimation (estimating a gradient), geometric progression of step size (taking a step), and boundary search (locating the boundary) operations to avoid collisions.

Estimating a Gradient. Gradient estimation works similarly to NES [25] but with a variance-reduced estimate over a different loss function as described below per the equations in the original paper:

$$\nabla_x^{est} L = \frac{1}{\zeta n} \sum_{i=1}^{n} (\phi_{x_{vic}}(x_t + \zeta u_i) - \overline{\phi_{x_{vic}}}) u_i, \tag{3}$$

where

$$\overline{\phi_{x_{vic}}} = \frac{1}{n} \sum_{i=1}^{n} \phi_{x_{vic}}(x_t + \zeta u_i), \tag{4}$$

$\phi_{x_{vic}}(x)$ is 1 if x is adversarial with respect to x_{vic} and -1 if it is not, u_i is a randomly sampled vector from the uniform distribution, and ζ is a small positive hyperparameter.

Modified: *To match the underlying attack distribution, we use a proposal distribution described by the uniform distribution on the surface of a sphere with radius ζ to sample the gradient estimation queries that avoid a collision. Then, we use OARS's adapt & resample to fine-tune ζ and generate n queries for gradient estimation.*

Taking a Step. For taking a step, vanilla HSJA [8] sets the initial step size to:

$$\lambda = \|x_t - x_{vic}\|_p / \sqrt{t} \tag{5}$$

where p represents the targeted threat model ℓ_p norm. HSJA then applies geometric progression, halving the step size until an adversarial example is found.

Modified: *We use a proposal distribution described by the uniform distribution on the surface of a sphere with radius λ to sample the smallest step queries that avoid a collision. Then, we use OARS's adapt & resample to sample the next step.*

Locating the Boundary. Vanilla HSJA [8] applies binary search between the post-step point and the original victim point to locate the boundary.

Modified: *Similar to the approach described in 3.1.4, we define a binary search with a termination distance and use a uniform proposal distribution parameterized by the upper bound. Then we use OARS's adapt & resample to proceed with the attack.*

3.2.4 ***OARS-QEBA (Hard-label).*** The changes for OARS-QEBA are the same as that of OARS-HSJA - the only algorithmic difference between vanilla QEBA [30] and vanilla HSJA [8] is using a lower dimensionality to sample from, and this algorithmic change does not necessitate an additional modification in how we apply OARS to HSJA.

3.2.5 ***OARS-SurFree (Hard-label).*** To create OARS-SurFree, we use OARS to modify two elements of vanilla SurFree [33] : magnitude adjustment of the polar coordinate representation (taking a step), and the final boundary location refinement between the proposed attack and the original victim sample (locating the boundary) to avoid collisions.

Taking a Step. Vanilla SurFree [33] relies on random search over directions parameterized by polar coordinates. In particular, given the original sample x_{vic}, the current attack iteration example x_t (with u defined as $x_{vic} - x_t$), SurFree tries to find new directions such that the boundary is closer. First, it uses Gram-Schmidt to sample a random vector v that is orthogonal to u. Then, SurFree computes a weighted addition of u and v described by the polar coordinate α.

Modified: *We use a proposal distribution described by the uniform distribution on the surface of a sphere with radius λ to sample the smallest step in the polar coordinate that avoids collision. Then, we use OARS's adapt & resample to sample the next step.*

Locating the Boundary. For locating the boundary, vanilla SurFree [33] applies a binary search between the final x_t and x_{vic}. Like OARS-HSJA, we apply OARS as described in Section 3.1.4.

3.2.6 ***OARS-Boundary (Hard-label).*** In order to create OARS-Boundary, we use OARS to modify vanilla Boundary's [6] to adapt and resample to evade collisions. Our application of OARS adjusts Boundary's adjustment of the hyperparameters η_δ (to control the distance to move in the random direction being searched - i.e., taking a step) and η_ϵ (to control the distance being moved back to the original point - i.e., locating the boundary).

Taking a Step. As vanilla Boundary [6] performs its random boundary walk, it iteratively operates two stages, with the first stage exploring k random orthogonal directions scaled by a factor of $0 < \eta_\delta < 1$, starting from an adversarial point. Boundary updates the value of η_δ using Trust Region methods: Compute the ratio of adversarial samples to the total samples; Increase η_δ if the ratio is too high; else, decrease it.

Modified: *We use a proposal distribution described by the uniform distribution on the surface of a sphere with radius λ to sample the smallest scaling factor that avoids collision. Then, we use OARS's adapt & resample to sample the next step.*

Locating the Boundary. In the second stage of vanilla Boundary [6], the algorithm then takes the adversarial points from the first stage and move back up to boundary towards the original victim sample η_ϵ. The value of η_ϵ is again adjusted with Trust Region methods, increasing ϵ if too many samples are adversarial and decreasing η_ϵ if there are too few adversarial samples to match a given threshold. We apply OARS to this stage by using the approach described in Section 3.1.4.

4 EXPERIMENTS

We evaluate our six OARS black-box attacks against four state-of-the-art SDMs: Blacklight [29], PIHA [11], IIoT-SDA [17], and OSD [9]. Below, we list three questions that our evaluation answers, along with a summary of the key takeaways:

Table 3: Overview of classification tasks, their datasets, classifiers, and test set accuracy.

Dataset	Classes	Model	Inp. Shape	Acc.
CIFAR10	10	ResNet-20	32x32x3	91.73%
ImageNet	1000	ResNet-152	224x224x3	78.31%
CelebaHQ	307	ResNet-152	256x256x3	89.55%
IIoT	2	5 Conv + FC	224x224x1	97.46%

• Are OARS attacks successful against vanilla SDMs?
We find our set of OARS attacks to be more successful than the existing query-blinding and standard attack baselines on the CIFAR10 [28], ImageNet [39], CelebaHQ [27], and IIoT Malware [17] datasets, which each defense suffering from at least one attack that achieves a 99% attack success rate. (Section 4.2)

• Do OARS attacks suffer when reconfiguring SDMs?
We consider different approaches SDMs might consider to thwart these attacks (e.g., different defense hyperparameters). We show that our attacks continue to adapt to these changes and successfully construct adversarial examples. (Section 4.3)

• What is the incurred attacker cost in the presence of SDM?
We report the number of needed queries for successful OARS attacks under different configurations. This includes queries incurred during both stages of OARS. Our findings indicate that these SDMs adaptations increase the number of queries required by the attacker. However, the cost of added queries is only $1-18 for online APIs. (Section 4.4)

4.1 Experimental Setup

Defenses and Classification Tasks. We test Blacklight [29] and PIHA [11] on three image classification tasks: CIFAR10 [28] and ImageNet [39] for object classification and CelebaHQ [27] for identity classification. We used ResNet [23] based models for these three datasets. We evaluate OSD [9] only on CIFAR10 as that is the only dataset it is available for. Finally, we evaluate IIoT-SDA [17] on the IIoT malware classification task [4] used in the original paper. Table 3 contains additional information about the datasets and the corresponding classification models.

Defense Configurations. In Section 4.2, we evaluate the vanilla SDMs under their default and originally proposed configurations. Specifically, Blacklight [29] utilizes a window size of 20 (for CIFAR10 [28], or 50 for ImageNet [39] and CelebaHQ [27]), a quantization step of 50, and a threshold of 0.5. PIHA [11] uses a block size of 7x7, and a threshold of 0.05. OSD [9] uses 50 nearest neighbors and a threshold of 1.44. IIoT [17] uses 11 nearest neighbors and a threshold of 0.21. We change these configurations in section 4.3 to evaluate the attacks in under modified versions of these SDMs.

Attack Configurations. We attack the above defenses with variants of the NES [25], Square [2], HSJA [8], QEBA [30], SurFree [33], and Boundary [6] attacks described in Sections 2.3. For each attack's vanilla hyperparameters, we employ the default values. NES and Square are ℓ_∞ score-based attacks, and HSJA, QEBA, SurFree, and Boundary are ℓ_2 hard-label attacks. Furthermore, NES, HSJA, and QEBA are targeted attacks; the remaining attacks are untargeted. For CIFAR10, ImageNet, and CelebaHQ, we use the standard ℓ_∞ and normalized ℓ_2 $\epsilon = 0.05$ perturbation budgets used in prior work [29]. For IIoT malware, we use the $\epsilon = 0.2$ budget employed by prior work [17]. Like in prior work [29], we also assume a query budget of 100k.

We run all attacks under two baseline configurations and three of our attack configurations. The first baseline is the standard configuration, which runs the attacks until success, query budget exhaustion, or a collision occurs. Note that this is the setting evaluated by prior work [29]. The second baseline is the query-blinding configuration, which builds upon the standard configuration by applying query-blinding [9]. We use the standard random affine transformation from prior work [29] for query-blinding, where each query is randomly rotated by up to 10° degrees, shifted by up to 10% horizontally/vertically, and zoomed by up to 10%. Again, attacks are run until success, budget exhaustion, or rejection. These transformations also apply to the data from the IIoT Malware dataset, which are represented in an 2d matrix format.

We evaluate three variants of our OARS attack configurations. The first two only apply OARS's resample strategy (only rejection sampling without fine-tuning the proposal distribution) to the standard and query blinding configurations. For a given attack, we enable the resample mechanisms described in Section 3 for *estimating a gradient*, *taking a step*, and *locating the boundary*. The third of our attack configurations includes the full recommended OARS configuration that enables both adapt and resample mechanisms (rejection sampling with proposal fine-tuning). Table 8 in Appendix B includes the detailed hyperparameters for all three configurations. We evaluate all five attack configurations against 100 samples, with target classes chosen uniformly at random for targeted attacks. For CIFAR10, we additionally report results on 1000 samples, included in Table 9 in Appendix A.

4.2 Our Adaptive Attacks vs. Existing SDMs

We now present and compare the attack success rates and query counts of our different attack configurations against the state-of-the-art rejection based defenses, Blacklight [29], PIHA [11], and IIoT-SDA [17] in Table 4. We report query counts averaged on only successful attacks. We first discuss the results of the baseline attack configurations and then the results of our adaptive OARS configurations. We conclude by evaluating OSD [9].

Baselines. Column 4 presents results of the standard configuration, as originally evaluated by the existing SDMs. As expected from prior work [29], we find that defenses are consistently able to thwart standard attacks as query collisions almost always occur. Notably, Blacklight [29] is robust with 0% ASR for all standard attacks across all datasets.

The few cases where standard attacks allow for non-zero (but low) ASR are for the PIHA [11] and IIoT-SDA [17] defenses (e.g., 14% for NES [25] against PIHA). The random search-based attack Square [2] produces low but non-zero ASR on all datasets against both PIHA and IIoT-SDA. Since this attack does not rely on gradient estimation and tends to converge very quickly (e.g., 2-120 queries), it has a lower chance of detection as compared to the slower-converging NES [25], HSJA [8], and QEBA [30] attacks. The gradient estimation procedure of the latter attacks repeatedly

Table 4: Our proposed adaptive attacks with adapt and resample outperform existing standard and query-blinding baselines. Results are presented in (ASR / query count) format. We find that with our adapt and resample techniques each dataset and defense combination suffers from at least one attack that achieves 99% or higher ASR. We also show that some attacks can show some improvement with just resampling mechanisms. Bold numbers are the best hard-label attacks, and bold italicized numbers are the best score-based attacks.

Dataset	Defense	Attack	Targeted	Baseline		Proposed		
				Standard	Query Blinding	Standard + Resample	Query Blinding + Resample	Adapt + Resample
				4	5	6	7	8
CIFAR10	Blacklight	NES	✓	0% / -	0% / -	0% / -	4% / 959	***99% / 1540***
		Square		0% / -	33% / 2	0% / -	34% / 11	93% / 218
		HSJA	✓	0% / -	0% / -	0% / -	0% / -	82% / 1615
		QEBA	✓	0% / -	0% / -	0% / -	0% / -	**98% / 1294**
		SurFree		0% / -	1% / 19	79% / 48	4% / 21	81% / 145
		Boundary		0% / -	0% / -	7% / 682	6% / 1449	**98% / 3302**
	PIHA	NES	✓	0% / -	0% / -	2% / 374	5% / 369	83% / 1646
		Square		29% / 3	35% / 2	90% / 101	38% / 3	***99% / 191***
		HSJA	✓	0% / -	0% / -	0% / -	0% / -	76% / 2811
		QEBA	✓	0% / -	0% / -	0% / -	1% / 14818	**95% / 1384**
		SurFree		0% / -	2% / 24	74% / 44	1% / 19	67% / 155
		Boundary		0% / -	0% / -	80% / 719	74% / 729	90% / 915
ImageNet	Blacklight	NES	✓	0% / -	0% / -	0% / -	0% / -	***100% / 10551***
		Square		0% / -	25% / 2	0% / -	25% / 2	84% / 173
		HSJA	✓	0% / -	0% / -	0% / -	0% / -	50% / 27620
		QEBA	✓	0% / -	0% / -	0% / -	0% / -	50% / 37924
		SurFree		0% / -	0% / -	71% / 204	0% / -	**93% / 1006**
		Boundary		0% / -	0% / -	28% / 1024	24% / 759	38% / 4262
	PIHA	NES	✓	14% / 9283	0% / -	34% / 8010	0% / -	***92% / 8578***
		Square		28% / 3	22% / 2	34% / 4	21% / 2	87% / 203
		HSJA	✓	0% / -	0% / -	0% / -	0% / -	91% / 29998
		QEBA	✓	0% / -	0% / -	0% / -	0% / -	96% / 22947
		SurFree		0% / -	0% / -	**100% / 555**	0% / -	**100% / 1892**
		Boundary		0% / -	0% / -	35% / 1980	32% / 1994	46% / 1278
CelebaHQ	Blacklight	NES	✓	0% / -	0% / -	0% / -	0% / -	***100% / 7719***
		Square		0% / -	14% / 2	0% / -	14% / 2	96% / 211
		HSJA	✓	0% / -	0% / -	0% / -	0% / -	77% / 31512
		QEBA	✓	0% / -	0% / -	0% / -	0% / -	93% / 8931
		SurFree		0% / -	0% / -	93% / 56	0% / -	**98% / 167**
		Boundary		0% / -	0% / -	51% / 697	54% / 1284	73% / 1974
	PIHA	NES	✓	39% / 7894	0% / -	74% / 7369	0% / -	97% / 7205
		Square		23% / 3	0% / -	33% / 5	14% / 2	***100% / 227***
		HSJA	✓	0% / -	0% / -	0%	0% / -	90% / 30934
		QEBA	✓	0% / -	0% / -	0% / -	0% / -	**100% / 6984**
		SurFree		0% / -	0% / -	**100% / 72**	0% / -	**100% / 175**
		Boundary		0% / -	0% / -	64% / 684	62% / 876	70% / 683
IIoT Malware	IIoT-SDA	NES	✓	10% / 52	4% / 25042	10% / 52	4% / 25042	97% / 3924
		Square		57% / 120	14% / 24	97% / 221	14% / 24	***100% / 615***
		HSJA	✓	0% / -	1% / 80468	35% / 1960	1% / 80468	**100% / 985**
		QEBA	✓	0% / -	1% / 48319	21% / 512	1% / 48319	**100% / 675**
		SurFree		91% / 210	0% / -	91% / 210	0% / -	98% / 455
		Boundary		0% / -	0% / -	11% / 395	13% / 414	30% / 398

queries a large number of nearby points to estimate gradients, adding to its query cost.

Column 5 presents results of the query blinding [9] configuration. Again, we find that the query-blinding attacks, as originally evaluated by existing defenses, fail except for a few cases of the

Square [2] attack. All other cases of other attacks have < 5% attack success rate. Manual inspection of the query-blinding attack failures corroborates our discussion in Section 2.5: arbitrary query transformations tend to disrupt the precision of the response, resulting in successful evasion at the cost of aimless "wandering." Overall, we find that neither standard nor query-blinding attacks are sufficient for attacking SDMs.

OARS. Columns 6-7 presents results for the *resample-only* configurations (rejection sampling without proposal distribution fine-tuning). Some attacks show moderate improvements. For example, the Square [2] and SurFree [33] attacks appear to benefit the most from the resample mechanisms (e.g., 100% for SurFree against PIHA [11] on ImageNet [39] and CelebaHQ [27]). The random search procedures employed instead of gradient estimation are likely to have contributed to this improvement. When combined with the fast convergence properties of a random search, resample mechanisms improve ASR, albeit modestly. However, there is room for more improvement: without adapting to the query collision distribution, the targeted attacks that employ gradient estimation (NES [25], HSJA [8], QEBA [30]) do not show significant improvement. Once again, adding query-blinding [9] to this configuration does not increase (and in many cases reduces) the ASR.

Column 8 then presents our full OARS attack configuration including both *adapt* and *resample* mechanisms that perform rejection sampling with proposal fine-tuning. All attacks show significant gains in ASR, with most attacks exceeding 80% (and in many cases 90%) ASR across all datasets. This column is the first to propose attacks that are widely successful in attacking existing SDMs, and suggests that both *adapt* and *resample* mechanisms are important for successful attacks.

OSD. As discussed in Section 3.1.5, if we can determine with a simple diagnostic test if the SDM in place is banning accounts and using a per-account query store, it may be more effective to simply run the standard attack under multiple (but relatively few) accounts.

We report the cost and ASR of what it would take to run the diagnostic tests in Section 3.1.5 then run the standard attacks. We find that we can compute these attacks in under 25 accounts, with Square [2] only requiring 2 and SurFree [33] requiring 3, even with the diagnostic test. Note that by using the first query used in the attack algorithm as the point x for the diagnostic test, we can reuse the query on account B to proceed with the attack, with either the standard or the OARS version.

4.3 Our Adaptive Attacks vs. Reconfigured SDMs

We now explore if SDMs can be easily reconfigured to be robust against OARS-enhanced attacks. Below we focus on Blacklight [29] as the most recent defense where we consider two reconfigurations that target its two key mechanisms: the similarity procedure and feature extraction.

Increasing the detection threshold. As our adaptive attacks use less similar queries, we evaluate raising the SDM's similarity threshold. This change should increase attack detection rate as more queries would collide. However, increasing Blacklight's similarity threshold drastically impacts its false positive rate, hurting its natural performance. Specifically, raising the threshold from the default 0.5 to 0.7 more than quadruples the false positive rate (from 0.2% to 0.9%) on ≈1 million images sampled from the tiny images dataset, which is the original database from which CIFAR10 is sampled [28]. Furthermore, even if the defense is willing to make this sacrifice, the ASR remains quite high (> 63% for all attacks). As an explanation, we plot the average pairwise distances between queries per threshold in Figure 3. The figure showcases the adaptive nature of our attacks: they automatically tune the distance between queries given the SDM settings. This observation follows from the

Table 5: We can attack OSD with few accounts by running our diagnostic tests to determine the defense is account banning based and then running the standard attacks. Results are presented in (ASR / query count) format. The cost of the diagnostic test (∼ 50 extra queries, 1 extra account) is included in the reported numbers. Square and SurFree can be completed in an average of 2 and 3 accounts respectively.

Attack	OSD	
	ASR	# Accounts
NES	100% / 861	18
Square	98% / 67	2
HSJA	100% / 1063	21
QEBA	100% / 948	19
SurFree	100% / 146	3
Boundary	100% / 680	14

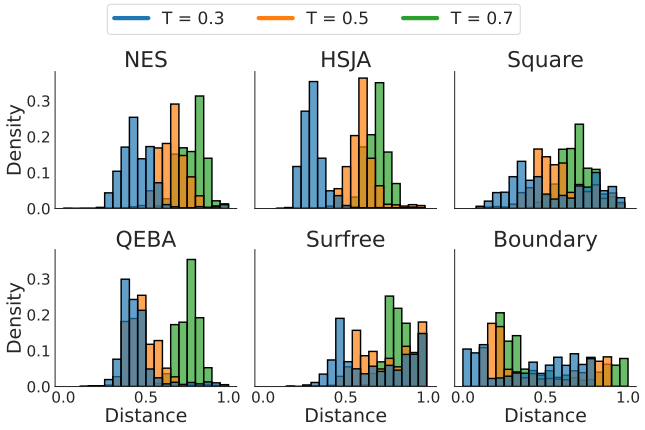


Figure 3: Our attacks automatically increase the distance between queries in response to the defense increasing its collision threshold. The horizontal axis shows pairwise distance between attack queries measured using the defense's distance function. Each histogram (blue, orange, green) corresponds to attacking an increasingly larger collision threshold used by the defense. Attacks are launched using the CIFAR10 dataset against Blacklight with a collision threshold $\in \{0.3, 0.5, 0.7\}$

Table 6: Our proposed adaptive attacks (adapt and resample mechanisms) continue to adapt and succeed with high ASR against SDMs even with adjusted hyperaparameter settings. Results are presented in (ASR / query count) format. Attacks are launched using the CIFAR10 dataset against Blacklight with 5 different configurations (1 default + 4 variations that adjust window size w and quantization step size q). The column headers represent (w,q).

Attack	Blacklight Alternate Configurations				
	(50,20)	(20,20)	(100,20)	(50,10)	(50,50)
NES	99% / 1540	97% / 1548	97% / 1548	96% / 1576	98% / 1585
Square	93% / 218	93% / 193	94% / 187	93% / 193	97% / 184
HSJA	82% / 1615	88% / 1636	89% / 1786	85% / 1721	91% / 1735
QEBA	86% / 1627	100% / 1047	96% / 1269	95% / 1405	100% / 1009
SurFree	81% / 145	99% / 186	81% / 144	80% / 142	97% / 189
Bound.	98% / 3302	94% / 1810	99% / 3302	99% / 3317	89% / 1785

design of the attacks that use rejection sampling. Further, our results suggest that natural performance drops off before adaptive attack is forced to use disruptive enough queries.

Adjusting the feature extractor's hyperparameters. We investigate whether changing Blacklight's [29] feature extractor hyperparameters improves robustness to our adaptive attacks. We change the window size and quantization step size for Blacklight, and set the threshold to meet the default 0.2% false positive rate. The results are shown in Table 6. As expected, we find that each of the tested settings are still vulnerable to our adaptive attack, with each setting susceptible to an attack that achieves 99% ASR. Attacks under all alternative Blacklight configurations achieve a 80% or greater ASR. These results suggest that creating truly robust SDMs is a challenging problem and will require more than simple tweaks of existing SDMs.

4.4 Attack Cost

We report the average number of queries needed by successful attacks in Table 4. In most cases, OARS attacks complete in less than 10k queries, with a few exceptions. Compared to attacks against an undefended model, the SDM can indeed raise the number of queries. For example, NES (score based; targeted) against CIFAR10 typically requires $\sim$ 900 queries for an undefended model, while OARS-NES takes $\sim$ 1.5k queries against a Blacklight-defended model. However, modern MLaaS platforms such as Clarifai, Google, and Microsoft, typically charge only \$1-\$1.50 per 1k queries. In such a case, the attacker would only pay an additional \$<1 to successfully generate an adversarial example against a defended model. This overhead depends on the threat model (soft/hard label; targeted/un-targeted) and varies among the different attacks. We observe the largest overhead for the weakest threat model (hard label; targeted) where a HSJA attacker would need to pay an additional $\sim$ \$18 to attack a ImageNet model.

Note that standard attacks are both expensive and perform poorly against SDM-defended models. Assuming a typical $100k$ query budget [29] they would cost > \$100 and typically fail to find adversarial examples against recent SDM models.

5 DISCUSSION

We now discuss possible SDM countermeasures to our OARS attacks, the applicability of our OARS attacks to other domains such as text, additional considerations on attacking per-account SDMs, limitations, and ethical considerations.

5.1 Countermeasures

While the majority of this work focused on understanding and evaluating existing SDMS (and their reconfigurations), we now consider alternative configurations. New SDMs can be designed by considering alternate choices for the feature extractor and action module. Against our proposed adaptive attacks, would alternate SDM designs work better?

Alternative Feature Extractors: A vast body of work exists on image feature extractors, outside the realm of adversarial defense [16]. We provide preliminary insight into understanding their viability as SDM feature extractors by evaluating the robust accuracy of SDM configurations with two popular perceptual feature extractors: PHash and Facebook's PDQ [14]. Specifically, we take Blacklight and only replace its feature extractor with PHash and PDQ. We find that these hash functions do not provide any gains in robustness, e.g., OARS-NES attacks are able to achieve 99% and 96% ASR respectively. This suggests that selecting an ideal feature extractor for SDMs is still a challenging problem for future work.

Alternative Action Function: Our adaptive attacks use multiple queries to extract information about the SDM. This is possible because the SDM itself leaks information while accepting/rejecting queries. One way for SDMs to avoid this is to respond with random labels or noisy probability scores. Such an action function could make it harder for an adversary to adapt. However, even so, an adaptive attack could estimate whether a response is random or not by analyzing the response distribution. Moreover, returning random responses could lead to unusual behaviour, i.e., querying the same image twice in a row against an SDM with random label responses could lead to a "free" adversarial example (with a perturbation $\epsilon = 0$).

Adaptive SDMs with Ensembling: One alternative for the defender is to try ensembling SDMs, in the hope that they can collectively prevent attacks. However, OARS does not make assumptions about the underlying defense and will treat the entire ensemble as a single black-box model, and should thus continue to be effective. We confirm this observation by evaluating vanilla and OARS attacks against an ensemble of Blacklight and PIHA on CIFAR10 in Table 7. We find that our attacks (without additional modification) are still successful, e.g., OARS-Square achieves an ASR of 95%. Furthermore, ensembling SDMs would raise the false positive rate of the system, suggesting that such an approach requires deeper investigation to be practical.

5.2 Applicability to Other Domains

Some SDMs claim to defend well against adversarial attacks in other domains such as text classification. For example, Blacklight claims perfect detection of the TextFooler synonym-subsitution adversarial attack on the IMDB dataset. TextFooler proceeds by (a) removing

Table 7: OARS attacks continue to be effective against ensembles of multiple SDMs. Results are presented in (ASR / query count) format. Attacks are launched using the CIFAR10 dataset against an ensemble of Blacklight and PIHA.

Attack	Standard	Adapt + Resample (OARS)
NES	0%	81% / 1627
Square	0%	95% / 205
HSJA	0%	67% / 3159
QEBA	0%	91% / 2078
SurFree	0%	63% / 137
Boundary	0%	89% / 2821

one word at a time and querying to identify an importance ordering, and (b) querying a series of synonym substitutions for the important words to elicit misclassification. To defend against this attack, Blacklight converts an input text to its embedding representation, and uses it to compute a similarity hash (similar to the image domain). We are able to confirm this observation — TextFooler's ASR is reduced from 100% to 0% against Blacklight on a BERT classifier for a test set of 100 samples.

Given the success of the OARS strategy in the image domain, one might consider its viability in other domains such as text. To this end, we now consider some preliminary results to show that it may indeed be applicable here as well. Given text sample x, we model the proposal distribution for all "similar" queries that also do no collide with x as the parametric edit-distance based sentence distribution. This distribution represents all sentences that are at some Levenshtein distance λ from x. Then, proposal p_λ can be adapted by performing oracle guided fine-tuning of λ. Finally, one can sample from fine-tuned distribution p_λ to construct attack queries — in practice, this amounts to insertion/removal of spaces and words from a query until edit distance is λ. We find that this approach is able to achieve a 78% ASR with an average of 20% words perturbed. This suggests that the OARS approach is likely relevant for other domains — we leave improvement of the ASR/reduction of the perturbed word count to future work.

5.3 Per-Account SDMs

One consideration that might make OSD [9] more suitable is a case where the assumption that accounts are easy to create in large quantities (through Sybil accounts [15, 46], for example) is no longer valid. However, we point out that due to the existence of improved attacks since the original paper (such as SurFree [33]), attacks can now be completed in as few as 2-3 accounts limiting this concern.

Another possibility is to consider reducing k (i.e., the number of queries required to cause an account ban). However, OSD's original analysis [9] suggests that this cannot be done without drastically sacrificing the false positive rate. In the future, if some defense that can use a lower k can be created, one can simply use the diagnostic tests described in Section 3.1.5 to compare the expected attack query costs of the attack and the expected overhead costs of OARS against the number of "free" queries per account and decide which strategy is better.

5.4 Generalizability

We note that the OARS approach of adapting proposal distributions and resampling substitute queries upon query collisions is general beyond the specific attack algorithms evaluated in the paper. OARS accommodates black-box attacks that comprise the three common elements discussed in Section 3.1. In Section 3.2, we show how to apply OARS to six black-box attacks that differ considerably (for example, NES [25] relies on finite differences to estimate the gradient, whereas Square performs random search along square perturbations). These six attacks thus provide blueprints for applying OARS to a wide range of new attacks. Specifically, one should start with identifying the pieces of the attack that map to the three common elements in Section 3.1 and apply the corresponding modifications to adapt the proposal distribution and resample as applicable. Note also that OARS is agnostic to the defense deployed, and should apply to future SDMs that leak information (See Section 5.1).

5.5 Overhead of OARS

Section 4.4 discusses the query overhead when running OARS. In general, this query overhead can depend upon two factors. First, the general trend suggests that more classes and higher input dimensionality incur a higher cost (this is also typical for standard attacks). Second, the query overhead appears to depend on the SDM deployed; we likely need more SDMs in existence to draw firmer conclusions on the exact relationship.

Another minor overhead is selecting the hyperparameters for the adaptation of the proposal distribution (e.g., upper/lower limits for adapting the variance of the Gaussian distribution). We emphasize that OARS uses the same hyperparameters regardless of the SDM (since the SDM is unknown). The hyperparameters only vary across datasets, which is consistent with standard black-box attacks. These attacks select hyperparameters that are dataset-specific to handle the different convergence rates based on the input dimensionality. Given an attack/dataset, we found it fairly simple to select hyperparameters by initializing them conservatively and observing convergence over 1-2 samples. Such hyperparameter selection typically took fewer than 15 minutes of experimentation and about 10 iterations of around 100 queries each, or about 1000 queries overall. This is a one-time cost per dataset.

5.6 Limitations

There are a few related attacks and defenses we do not consider in this work. Firstly, we do not consider transfer attacks. Although transfer attacks can be crafted with the assistance of techniques such as model stealing [26], we follow prior work on SDMs and consider transfer attacks to be an orthogonal problem — transfer attack defenses already exist [42] and can be combined with stateful systems to build a more complete defense.

Secondly, we do not consider alternative approaches to detecting black-box attacks such as Adversarial Attack on Attackers (AAA) [10] and AdvMind [36]. AAA is a post-processing attack that attempts to modify the logits loss curve to locally point in the incorrect attack direction in a periodic fashion, misguiding score-based query attacks from a successful attack. AdvMind is a detection model that infers the intent of an adversary and detect attacks.

While these approaches both aim to thwart black-box attacks, they are orthogonal to SDMs that aim to detect similar queries.

5.7 Ethical Considerations

While our paper exposes new attack strategies that could then be used to attack real-world systems deploying SDMs, it is important for developers to have the system, software, and algorithmic tools necessary to truly understand the potential vulnerabilities and true robustness of any SDM that may be in consideration for future deployment. We note that query-based black-box attacks already exist and have been used to attack real-world systems already [30]. Our hope is that, analogously to the white-box setting, our work encourages stronger evaluation of SDMs before we reach a point where they are being falsely relied on to solve the robustness problem in real-world deployments.

6 CONCLUSION

This paper proposes OARS, an adaptive black-box attack strategy that significantly increases the attack success rate against four state-of-the-art SDMs. Our key insight is that SDMs leak information about the similarity-detection procedure and its parameters, enabling an attacker to adapt its queries to evade collisions. Our work shows that these SDMs are not as truly robust as previously believed, and provides a new benchmark to test future SDMs.

ACKNOWLEDGEMENTS

This material is based upon work supported by DARPA under agreement number 885000, National Science Foundation Grant No. 2039445, and National Science Foundation Graduate Research Fellowship Grant No. DGE 1841052. Any opinion, findings, and conclusions or recommendations expressed in this material are those of the authors(s) and do not necessarily reflect the views of our research sponsors.

REFERENCES

[1] Amazon. [n. d.]. Amazon Rekognition: Automate your image recognition and video analysis with machine learning. https://aws.amazon.com/rekognition/

[2] Maksym Andriushchenko, Francesco Croce, Nicolas Flammarion, and Matthias Hein. 2020. Square attack: a query-efficient black-box adversarial attack via random search. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXIII*. Springer, 484–501.

[3] Anish Athalye, Nicholas Carlini, and David Wagner. 2018. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *International conference on machine learning*. PMLR, 274–283.

[4] Amin Azmoodeh, Ali Dehghantanha, and Kim-Kwang Raymond Choo. 2018. Robust malware detection for internet of (battlefield) things devices using deep eigenspace learning. *IEEE transactions on sustainable computing* 4, 1 (2018), 88–95.

[5] Sean Bell and Kavita Bala. 2015. Learning visual similarity for product design with convolutional neural networks. *ACM transactions on graphics (TOG)* 34, 4 (2015), 1–10.

[6] Wieland Brendel, Jonas Rauber, and Matthias Bethge. 2017. Decision-based adversarial attacks: Reliable attacks against black-box machine learning models. *arXiv preprint arXiv:1712.04248* (2017).

[7] Tom B Brown, Dandelion Mané, Aurko Roy, Martín Abadi, and Justin Gilmer. 2017. Adversarial patch. *arXiv preprint arXiv:1712.09665* (2017).

[8] Jianbo Chen, Michael I Jordan, and Martin J Wainwright. 2020. Hopskipjumpattack: A query-efficient decision-based attack. In *2020 ieee symposium on security and privacy (sp)*. IEEE, 1277–1294.

[9] Steven Chen, Nicholas Carlini, and David Wagner. 2020. Stateful detection of black-box adversarial attacks. In *Proceedings of the 1st ACM Workshop on Security and Privacy on Artificial Intelligence*. 30–39.

[10] Sizhe Chen, Zhehao Huang, Qinghua Tao, Yingwen Wu, Cihang Xie, and Xiaolin Huang. 2022. Adversarial Attack on Attackers: Post-Process to Mitigate Black-Box Score-Based Query Attacks. *arXiv preprint arXiv:2205.12134* (2022).

[11] Seok-Hwan Choi, Jinmyeong Shin, and Yoon-Ho Choi. 2023. PIHA: Detection method using perceptual image hashing against query-based adversarial attacks. *Future Generation Computer Systems* (2023).

[12] Clarifai. [n. d.]. The world's AI: Clarifai Computer Vision AI and Machine Learning Platform. https://www.clarifai.com/

[13] Francesco Croce and Matthias Hein. 2020. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *International conference on machine learning*. PMLR, 2206–2216.

[14] Janis Dalins, Campbell Wilson, and Douglas Boudry. 2019. PDQ & TMK+ PDQF–A Test Drive of Facebook's Perceptual Hashing Algorithms. *arXiv e-prints* (2019), arXiv–1912.

[15] John R Douceur. 2002. The sybil attack. In *Peer-to-Peer Systems: First InternationalWorkshop, IPTPS 2002 Cambridge, MA, USA, March 7–8, 2002 Revised Papers 1*. Springer, 251–260.

[16] Ling Du, Anthony TS Ho, and Runmin Cong. 2020. Perceptual hashing for image authentication: A survey. *Signal Processing: Image Communication* 81 (2020), 115713.

[17] Bardia Esmaeili, Amin Azmoodeh, Ali Dehghantanha, Hadis Karimipour, Behrouz Zolfaghari, and Mohammad Hammoudeh. 2022. IIoT deep malware threat hunting: from adversarial example detection to adversarial scenario detection. *IEEE Transactions on Industrial Informatics* 18, 12 (2022), 8477–8486.

[18] Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Chaowei Xiao, Atul Prakash, Tadayoshi Kohno, and Dawn Song. 2018. Robust physical-world attacks on deep learning visual classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1625–1634.

[19] Ryan Feng, Neal Mangaokar, Jiefeng Chen, Earlence Fernandes, Somesh Jha, and Atul Prakash. 2022. GRAPHITE: Generating Automatic Physical Examples for Machine-Learning Attacks on Computer Vision Systems. In *2022 IEEE 7th European Symposium on Security and Privacy (EuroS&P)*. IEEE, 664–683.

[20] Andreas Geiger, Philip Lenz, and Raquel Urtasun. 2012. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE conference on computer vision and pattern recognition*. IEEE, 3354–3361.

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

[22] Varun Gulshan, Lily Peng, Marc Coram, Martin C Stumpe, Derek Wu, Arunachalam Narayanaswamy, Subhashini Venugopalan, Kasumi Widner, Tom Madams, Jorge Cuadros, et al. 2016. Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs. *Jama* 316, 22 (2016), 2402–2410.

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

[24] Ashish Hooda, Neal Mangaokar, Ryan Feng, Kassem Fawaz, Somesh Jha, and Atul Prakash. 2022. Towards Adversarially Robust Deepfake Detection: An Ensemble Approach. arXiv:2202.05687 [cs.LG]

[25] Andrew Ilyas, Logan Engstrom, Anish Athalye, and Jessy Lin. 2018. Black-box adversarial attacks with limited queries and information. In *International conference on machine learning*. PMLR, 2137–2146.

[26] Mika Juuti, Sebastian Szyller, Samuel Marchal, and N Asokan. 2019. PRADA: protecting against DNN model stealing attacks. In *2019 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 512–527.

[27] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. 2018. Progressive Growing of GANs for Improved Quality, Stability, and Variation. In *International Conference on Learning Representations*.

[28] Alex Krizhevsky, Geoffrey Hinton, et al. 2009. Learning multiple layers of features from tiny images. (2009).

[29] Huiying Li, Shawn Shan, Emily Wenger, Jiayun Zhang, Haitao Zheng, and Ben Y Zhao. 2022. Blacklight: Scalable Defense for Neural Networks against {Query-Based}{Black-Box} Attacks. In *31st USENIX Security Symposium (USENIX Security 22)*. 2117–2134.

[30] Huichen Li, Xiaojun Xu, Xiaolu Zhang, Shuang Yang, and Bo Li. 2020. Qeba: Query-efficient boundary-based blackbox attack. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 1221–1230.

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

[32] Prasanta Chandra Mahalanobis. 1936. On the generalized distance in statistics. National Institute of Science of India.

[33] Thibault Maho, Teddy Furon, and Erwan Le Merrer. 2021. Surfree: a fast surrogate-free black-box attack. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 10430–10439.

[34] Seungyong Moon, Gaon An, and Hyun Oh Song. 2019. Parsimonious black-box adversarial attacks via efficient combinatorial optimization. In *International Conference on Machine Learning*. PMLR, 4636–4645.

[35] Timo Ojala, Matti Pietikäinen, and David Harwood. 1996. A comparative study of texture measures with classification based on featured distributions. *Pattern recognition* 29, 1 (1996), 51–59.
[36] Ren Pang, Xinyang Zhang, Shouling Ji, Xiapu Luo, and Ting Wang. 2020. Advmind: Inferring adversary intent of black-box attacks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1899–1907.
[37] Jiameng Pu, Neal Mangaokar, Lauren Kelly, Parantapa Bhattacharya, Kavya Sundaram, Mobin Javed, Bolun Wang, and Bimal Viswanath. 2021. Deepfake videos in the wild: Analysis and detection. In *Proceedings of the Web Conference 2021*. 981–992.
[38] Plate Recognizer. 2022. Automatic license plate recognition - high accuracy ALPR. https://platerecognizer.com/
[39] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. 2015. Imagenet large scale visual recognition challenge. *International journal of computer vision* 115 (2015), 211–252.
[40] Yi Sun, Xiaogang Wang, and Xiaoou Tang. 2014. Deep learning face representation from predicting 10,000 classes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1891–1898.
[41] Florian Tramer, Nicholas Carlini, Wieland Brendel, and Aleksander Madry. 2020. On adaptive attacks to adversarial example defenses. *Advances in neural information processing systems* 33 (2020), 1633–1645.
[42] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. 2017. Ensemble adversarial training: Attacks and defenses. *arXiv preprint arXiv:1705.07204* (2017).
[43] Dayong Wang, Aditya Khosla, Rishab Gargeya, Humayun Irshad, and Andrew H Beck. 2016. Deep learning for identifying metastatic breast cancer. *arXiv preprint arXiv:1606.05718* (2016).
[44] Daan Wierstra, Tom Schaul, Tobias Glasmachers, Yi Sun, Jan Peters, and Jürgen Schmidhuber. 2014. Natural evolution strategies. *The Journal of Machine Learning Research* 15, 1 (2014), 949–980.
[45] Ziang Yan, Yiwen Guo, Jian Liang, and Changshui Zhang. 2021. Policy-driven attack: learning to query for hard-label black-box adversarial examples. In *International Conference on Learning Representations*.
[46] Zhi Yang, Christo Wilson, Xiao Wang, Tingting Gao, Ben Y Zhao, and Yafei Dai. 2014. Uncovering social network sybils in the wild. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 8, 1 (2014), 1–29.

A ADDITIONAL EVALUATION

As discussed in Section 4.2, Table 9 presents extended results of OARS against a larger sample of 1000 CIFAR10 images against Blacklight.

B ATTACK HYPERPARAMETERS

As discussed in Section 4.2, Table 8 presents hyperparameters for OARS adapt and resample attacks presented in Table 4.

Table 9: Extended results for OARS attacks with adapt and resample. Results are computed over 1000 images on CIFAR10 against Blacklight.

Attack	Standard	Adapt + Resample
NES	0%	96% / 1585
Square	0%	93% / 206
HSJA	0%	86% / 1573
QEBA	0%	97% / 1324
SurFree	0%	83% / 149
Boundary	0%	99% / 2819

Dataset	Attack	ge_{tries}	σ				$steps_{tries}$	$step$			
			$stps$	sam	cr	$[\sigma_{lo}, \sigma_{hi}]$		$stps$	sam	cr	$[step_{lo}, step_{hi}]$
CIFAR	**NES**	5	10	20	0	[0.05, 0.5]	20	10	20	0.005	[0.1,0.1]
	Square	-	-	-	-	-	300	10	5 10	0 0.5	[10,100], [1,32]
	HSJA	20	10	20	0.05	[1,2]	5	-	-	-	-
	QEBA	20	10	20	0.05	[1,2]	5	-	-	-	-
	SurFree	-	-	-	-	-	100	5	20	0.05	[5,50]
	Boundary	-	-	-	-	-	-	-	-	-	-
ImageNet	**NES**	5	10	20	0.05	[0.01, 0.5]	5	10	20	0.13	[0.00005, 0.1]
	Square	-	-	-	-	-	300	3 10	5 10	0 0.5	[50,100],[1,100]
	HSJA	20	10	20	0.05	[1.5,5]	5	-	-	-	-
	QEBA	20	10	20	0.05	[1.5,5]	5	-	-	-	-
	SurFree	-	-	-	-	-	100	5	20	0.05	[5,50]
	Boundary	-	-	-	-	-	-	-	-	-	-
CelebaHQ	**NES**	5	10	20	0.05	[0.01, 0.5]	5	10	20	0.13	[0.00005, 0.1]
	Square	-	-	-	-	-	300	3 10	5 10	0 0.5	[50,100],[1,100]
	HSJA	20	10	20	0.05	[0.5,5]	5	-	-	-	-
	QEBA	20	10	20	0.05	[0.5,5]	5	-	-	-	-
	SurFree	-	-	-	-	-	100	5	20	0.05	[5,50]
	Boundary	-	-	-	-	-	-	-	-	-	-
IIoT	**NES**	5	10	20	0.1	[0.00001,0.5]	5	10	20	0.05	[0.00005,1]
	Square	-	-	-	-	-	5	10	5 10	0 0.5	[10,100], [1,32]
	HSJA	1	20	20	0.05	[0.001,20]	5	20	20	0.05	[0.00005,1]
	QEBA	1	20	20	0.05	[0.001,20]	5	20	20	0.05	[0.00005,1]
	SurFree	-	-	-	-	-	10	10	20	0.05	[1,50]
	Boundary	-	-	-	-	-	-	-	-	-	-

Table 8: OARS adapt and resample hyperparameters for all attacks and datasets. Hyperparameters are selected on a per-attack basis, i.e., lo and hi values by observing valid ranges of values that typically work with the standard attack on undefended models, and $stps$, sam chosen to obtain a stable estimate of cr which is generally set to be low.