# Evading Black-box Classifiers Without Breaking Eggs

Edoardo Debenedetti[1] Nicholas Carlini[2] Florian Tramèr[1]
[1]*ETH Zurich* [2]*Google DeepMind*

***Abstract*—Decision-based evasion attacks repeatedly query a black-box classifier to generate adversarial examples. Prior work measures the cost of such attacks by the total number of queries made to the classifier. We argue this metric is incomplete. Many security-critical machine learning systems aim to weed out "bad" data (e.g., malware, harmful content, etc). Queries to such systems carry a fundamentally *asymmetric cost*: *flagged queries*, i.e., queries detected as "bad" by the classifier come at a higher cost because they trigger additional security filters, e.g., usage throttling or account suspension. Yet, we find that existing decision-based attacks issue a large number of queries that would get flagged by a security-critical system, which likely renders them ineffective against such systems. We then design new attacks that reduce the number of flagged queries by 1.5–7.3×. While some of our attacks achieve this improvement at the cost of only a moderate increase in total (including non-flagged) queries, other attacks require significantly more total queries than prior attacks. We thus pose it as an open problem to build black-box attacks that are more effective under realistic cost metrics.**

***Index Terms*—security, threat models, black-box adversarial examples, decision-based attacks**

## I. Introduction

Adversarial examples [34, 2] are a security risk for machine learning (ML) models that interact with malicious actors. For example, an attacker could use adversarial examples to post undesired content to the Web while bypassing ML filtering mechanisms [35, 27, 30]. In such security-critical uses of ML, the attacker often only has *black-box* access to the ML model's decisions.

*Decision-based attacks* [3] generate adversarial examples in black-box settings by repeatedly querying the model and observing only the output decision on perturbed inputs. The original Boundary Attack of Brendel et al. [3] required over 100,000 model queries to reliably find small adversarial perturbations. Subsequent work [10, 11, 6, 7, 20] has optimized for this metric of "total number of model queries", and reduced it by 1–3 orders of magnitude.

**We argue this metric fails to reflect the true cost of querying a security-critical ML system**. Such systems typically aim to detect "bad" data, such as malware, harmful content or malicious traffic. Queries with benign data (e.g., a selfie uploaded to social media) carry little cost; in contrast, bad data that is detected and flagged by the system (e.g., offensive content) triggers additional security measures that carry a high cost for the attacker—up to account termination [17, 23]. Thus, we argue that black-box attacks should strive to be *stealthy*, by minimizing the number of "bad" queries that are flagged by the ML system—rather than the total number of queries (both bad and benign).

Edoardo Debenedetti is supported by armasuisse Science and Technology.

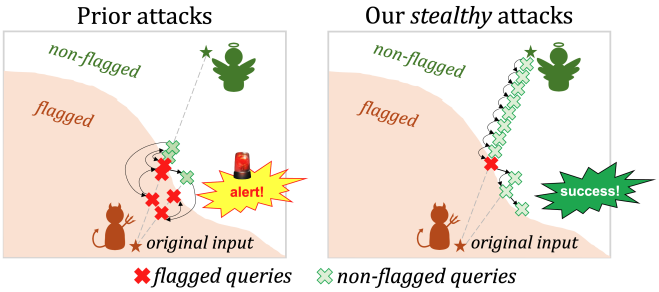



Fig. 1: Existing decision-based attacks (left) make many "flagged" queries, that get classified into the class that the attacker aims to evade (e.g., NSFW content). In security-critical applications, such flagged queries would trigger additional security mechanisms, thus raising the cost of the attack. Our *stealthy* attacks (right) trade-off flagged queries for non-flagged ones, to find adversarial examples without raising security alerts.

We find that existing attacks are not stealthy: over 50% of the queries they make are flagged. We then show how to drastically reduce the number of flagged queries for a class of attacks that measure distances to the model's boundary along random directions (e.g., Opt [10], Sign-Opt [11] and RayS [7]). Inspired by the famous "egg-dropping problem"[1], we design variants of these attacks (for both the $\ell_\infty$ and $\ell_2$ norms) that trade-off flagged queries for non-flagged ones.

We evaluate our attacks on three classification tasks: ImageNet, a binary dog vs. not-dog classification task on ImageNet, and NSFW content detection [32]. Our stealthy attacks reduce the number of flagged queries of the original attacks by 1.5–7.3×. Notably, our stealthy variant of the $\ell_\infty$ RayS attack reduces flagged queries by 2.1–2.5× over RayS and 6–17× over HopSkipJump, while making 2.1–3.4× more benign, non-flagged queries. We then use our stealthy RayS attack to evade a commercial black-box NSFW image detector, with 2.2× fewer flagged queries than the original RayS attack.

For $\ell_2$-attacks, our stealthy attacks similarly reduce the number of flagged queries compared to prior attacks. Notably,

[1]The egg-dropping problem is a mathematical exercise that asks to find the maximal storey in a building from which an egg can be dropped without breaking while incurring at most $k \geq 1$ broken eggs: https://brilliant.org/wiki/egg-dropping.

our stealthy variant of the OPT attack outperforms SIGN-OPT and HOPSKIPJUMP in terms of *flagged* queries, despite the two latter attacks issuing fewer queries *in total*. However, our stealthy attacks also incur a much higher cost in non-flagged queries (and thus in the number of total queries issued by the attack). Concretely, our most stealthy $\ell_2$ attack (based on HOPSKIPJUMP) reduces flagged queries by a factor 1.5–1.8×, but incurs a large increase in non-flagged queries (350–1,400×). Our stealthy $\ell_2$ attacks are thus likely only cost-effective in scenarios where flagged queries carry a significantly higher cost ($\approx$ 1,000×) than non-flagged ones (this cost ratio may be realistic when flagged queries are likely to trigger costly measures such as account termination).

Overall, our results suggest that many decision-based attacks are far from stealthy and that stealthier attacks are viable if the cost of flagged queries far outweighs that of non-flagged queries (especially for $\ell_2$ attacks). We thus recommend that future decision-based attacks account for asymmetric query costs, to better reflect the true cost of deploying such attacks against real security-critical systems.

## II. DECISION-BASED ATTACKS

Given a classifier $f : [0,1]^d \to \mathcal{Y}$ and input $(x, y)$, an (untargeted) adversarial example $\hat{x}$ is an input close to $x$ that is misclassified, i.e., $f(\hat{x}) \neq y$ and $\|\hat{x} - x\|_p \leq \epsilon$ for some $\ell_p$ norm and threshold $\epsilon$.

A decision-based attack gets oracle access to the model $f$. The attacker can query the model on arbitrary inputs $x \in [0,1]^d$ to obtain the class label $y \leftarrow f(x)$. Existing decision-based attacks aim to minimize the **total number of queries** made to the model $f$ before the attack succeeds.

**Applications.** Decision-based attacks [3] were designed for black-box ML systems that only return model decisions (e.g., an ML model that filters social media content). Such attacks are also applicable when an attacker has physical access to a model guarded by hardware protections, e.g., a phone's authentication mechanism, or a self-driving system. Decision-based attacks are also commonly used to evaluate the robustness of *white-box* models, when computing gradients is hard [3, 36].

In this paper, we are interested in the first two scenarios, where decision-based attacks are used against black-box ML security systems. In particular, we assume that these security systems monitor and log user queries, and can throttle or disable an attacker's access to the system.

## III. ASYMMETRIC QUERY COSTS

Existing decision-based attacks optimize for the *total* number of model queries. This is reasonable if the attacker's primary cost is incurred by queries to the model, and this cost is *uniform* across queries (e.g. if the attacker has to pay a fixed service fee for each query).

However, we argue that query costs are rarely uniform in practical security-critical systems. This is because, in such systems, the goal of an ML model is usually to detect and flag "bad" data (e.g., malware, harmful content, malicious traffic, etc). The costs incurred by querying such a model are highly asymmetric. Querying the model with "good" data that does not get flagged is expected, comes with no additional overhead, and is thus cheap. Whereas querying the model with "bad" data is unexpected, triggers additional security measures and filters, and thus places a much higher cost on the attacker.

We first formalize the notion of "flagged" and "non-flagged" queries. We assume that the ML system uses a classifier $f : [0,1]^d \to \mathcal{Y}$ to filter out bad content, and that some subset of the output classes $\mathcal{Y}_{\text{bad}} \subset \mathcal{Y}$ correspond to bad data (e.g., social media content that is NSFW, offensive, etc.) We then define *flagged* and *non-flagged* queries (or inputs) as follows:

**Flagged queries**

A query $x$ is *flagged* if the target model $f$ labels it as bad so that it is filtered out, i.e., $f(x) \in \mathcal{Y}_{\text{bad}}$.

Conversely:

**Non-flagged queries**

A query $x$ is *non-flagged* if the target model classifies it as benign, i.e., $f(x) \notin \mathcal{Y}_{\text{bad}}$. That is, if it is not *flagged*.

It is important to note that a query $x$ being flagged or non-flagged is solely a property of the classifier $f$ and *not* of the input's ground truth label. For example, a benign image (e.g., a cute puppy) can be flagged; this is a *false-positive* of the classifier. Conversely, a **successful** adversarial example for an NSFW image will—by definition—be **non-flagged** even though it is objectively bad; this is thus a *false-negative* of the classifier.

In an evasion attack, the attacker is given an input $(x, y)$ that is (objectively) bad, i.e., $y \in \mathcal{Y}_{\text{bad}}$, and their goal is to find an adversarial example $\hat{x}$ that is not flagged by the system, i.e., $f(\hat{x}) \notin \mathcal{Y}_{\text{bad}}$. All queries made by the attacker to the model $f$ carry a base cost $c_0$, due to data processing, network bandwidth, disk storage, or throttling if the attacker makes too many queries. This base cost is typically very low: e.g., Facebook users can upload 1,000 images at once in an album [13]. However, for queries $x$ that are flagged as inappropriate (i.e., $f(x) \in \mathcal{Y}_{\text{bad}}$), the cost $c_{\text{flagged}}$ incurred by the attacker is much larger. Their account could be suspended or banned, their IP blacklisted, etc. While these restrictions can be circumvented (e.g., by buying multiple accounts [5]), this places a significantly higher cost on queries flagged as flagged, i.e., $c_{\text{flagged}} \gg c_0$.

We thus argue that decision-based attacks should strive to minimize the following cost:

$$\textit{minimize} \quad \texttt{cost} \coloneqq Q_{\text{total}} \cdot c_0 + Q_{\text{flagged}} \cdot c_{\text{flagged}} \,, \qquad (1)$$

where $Q_{\text{flagged}}$ is the number of flagged model queries ($f(x) \in \mathcal{Y}_{\text{flagged}}$), $Q_{\text{total}}$ is the total number of queries—including flagged ones—and $c_{\text{flagged}} \gg c_0$. We call attacks that minimize this asymmetric cost *stealthy*.

**Example: evading NSFW content detection.** To make the above discussion more concrete, consider the example of an

TABLE I: Median number of queries for each attack to reach a median $\ell_2$ distance of 10 and median $\ell_\infty$ distance of $^8/_{255}$ on untargeted ImageNet. We report the total number of attack queries $Q_{\text{total}}$, and of "flagged" queries $Q_{\text{flagged}}$ (queries that get classified as the class that the attacker wants to evade).

| Norm | Attack | Total Queries ($Q_{\text{total}}$) | Flagged Queries ($Q_{\text{flagged}}$) |
|---|---|---|---|
| $\ell_2$ | OPT | 9,731 | 4,975 (51%) |
| | BOUNDARY | 4,555 | 3,843 (84%) |
| | SIGN-OPT | 2,873 | 1,528 (53%) |
| | HOPSKIPJUMP | 1,752 | 953 (54%) |
| $\ell_\infty$ | HOPSKIPJUMP | 3,591 | 1,789 (50%) |
| | RAYS | 328 | 244 (74%) |

attacker who tries to upload inappropriate content to a social media website. Every uploaded image passes through an ML model that flags inappropriate content. Here, a *flagged* query is **any** query to the system that the NSFW detector classifies as NSFW, and which will thus be filtered out.

Uploading content that is non-flagged carries little cost: the social media platform will simply post the picture, and only apply rate limits once the user uploads a very large number of non-flagged content. But if a query *is* flagged as inappropriate, the system is likely to block the contents and take further costly actions (e.g., account throttling, suspension, or termination). We thus have that $c_{\text{flagged}} \gg c_0$ (we further discuss how to estimate these costs below and in Section VI).

**Existing attacks are not designed to be stealthy.** No existing black-box attack considers such asymmetric query costs. As a result, these attacks issue a large number of flagged queries. We illustrate this with an untargeted attack on ImageNet.[2] In Table I, we show the number of total queries $Q_{\text{total}}$ and flagged queries $Q_{\text{flagged}}$ made by various $\ell_2$ and $\ell_\infty$ decision-based attacks on a ResNet-50 classifier. In all cases, half or more of the attacker's queries are "flagged" (i.e., they get the class label that was to be evaded). Despite differences in the fraction of flagged queries for each attack, attacks that make fewer total queries also make fewer flagged queries. But this begs the question of whether we could design attacks that issue far fewer flagged queries in total. The remainder of this paper answers this question.

**Selecting the values of $c_0$ and $c_{\text{flagged}}$.** The true cost of a query (whether non-flagged or flagged) may be hard to estimate, and can vary between applications. As a result, we recommend that black-box attack evaluations report both the value of $Q_{\text{total}}$ and $Q_{\text{flagged}}$ so that the attack cost can be calculated for any domain-specific values of $c_0$ and $c_{\text{flagged}}$.

[2]ImageNet is not a security-critical task, and thus most content is not "bad". We use ImageNet here because prior attacks were designed to work well on it. To mimic a security-critical evasion attempt, we set the class to be evaded as "flagged" and all other classes as "non-flagged". That is, for an input $(x, y)$ we set $\mathcal{Y}_{\text{flagged}} = \{y\}$ and the attacker's goal is to find an adversarial example $\hat{x}$ such that $f(\hat{x}) \neq y$ while avoiding making queries labeled as $y$.

In this paper, we often make the simplifying assumption that $c_0 = 0, c_{\text{flagged}} = 1$, a special case that approximates the attack cost when $c_{\text{flagged}} \gg c_0$. In this special case, the attacker solely aims to minimize flagged queries, possibly at the expense of a large increase in total queries. We will, however, also consider what the trade-offs look like in real-world applications in Section VI.

## IV. DESIGNING STEALTHY DECISION-BASED ATTACKS

We explore the design space of stealthy decision-based attacks, which minimize the total number of flagged queries made to the model.

One possibility is simply to design a *better* decision-based attack, that makes fewer *total* queries. As we see from Table I, this is how prior work has implicitly minimized asymmetric attack costs so far. We take a different approach, and design attacks that explicitly *trade-off* flagged queries for non-flagged ones.

In Section IV-A, we first review how existing decision-based attacks work, and distill some common sub-routines that help us understand how these attacks spend their queries (either flagged or non-flagged ones). In Section IV-B, we then show how to design stealthy variants of these sub-routines, and in Section IV-C we describe the resulting stealthy attacks that we introduce. In Section IV-D we formally prove that our stealthy attack techniques are more efficient (in terms of flagged queries) than prior attacks.

### A. How do Decision-based Attacks Work?

Most decision-based attacks follow the same blueprint [15]. For an input $(x, y)$, the attacks first pick an *adversarial direction* $\theta \in [0, 1]^d$ and find the $\ell_p$ distance to the model's decision boundary from $x$ along the direction $\theta$. They then iteratively perturb $\theta$ to minimize the boundary distance along the new direction. Each iteration of the attacks can be divided into three phases:

- `projBoundary`: given the original input $(x, y)$ and a search direction $\theta$, this phase finds a point $x_b$ that lies on the model's decision boundary along the line $x + \alpha \cdot \theta/\|\theta\|$., and returns the $\ell_p$ distance between $x$ and $x_b$, i.e., $\texttt{dist} \leftarrow \|x - x_b\|_p$.
- `updateDir`: This phase searches for an update direction $\delta$ to be applied to the search direction $\theta$.
- `stepSize`: This phase selects a step-size $\alpha$ for an update to the search direction $\theta$.

For our purposes, it will be helpful to further decompose each of these three phases into two fundamental subroutines:

- $\texttt{getDist}(x, \theta, p) \rightarrow \mathbb{R}^+$: this routine computes the distance (in $\ell_p$ norm) from $x$ to the decision boundary along the direction $\theta$. Most attacks do this by performing a binary search between $x$ and a misclassified point $\hat{x}$ in the direction $\theta$, up to some numerical tolerance $\eta$.
- $\texttt{checkAdv}(x, \theta', \texttt{dist}, y) \rightarrow \{-1, 1\}$: this routine uses a single query to check if the point at distance `dist` in direction $\theta'$ is misclassified, i.e., it returns 1 if $f(x + \texttt{dist} \cdot \theta'/\|\theta'\|_p) \neq y$.

TABLE II: Queries issued by different decision-based attacks in a single attack iteration. We distinguish between checkAdv queries that check whether some arbitrary direction yields a misclassification and getDist queries that issue multiple calls to the model to measure the distance to the model boundary along some direction. The hyper-parameter $n$ is the number of times a routine is called for estimating the geometry of the model's decision boundary. The variable $m$ is the average number of step-size searches conducted in one iteration of OPT and SIGN-OPT.

| | Attack Phase | | | Total |
|---|---|---|---|---|
| **Attack** | **projBoundary** | **updateDir** | **stepSize** | |
| BOUNDARY | checkAdv $\cdot\, n$ | – | checkAdv | checkAdv $\cdot\,(n+1)$ |
| OPT | getDist | getDist $\cdot\, n$ | getDist $\cdot\, m$ | getDist $\cdot\,(n+m+1)$ |
| SIGN-OPT | getDist | checkAdv $\cdot\, n$ | getDist $\cdot\, m$ | checkAdv $\cdot\, n$ + getDist $\cdot\,(m+1)$ |
| HOPSKIPJUMP | getDist | checkAdv $\cdot\, n$ | getDist | checkAdv $\cdot\, n$ + getDist $\cdot\, 2$ |
| RAYS | getDist | – | checkAdv | checkAdv + getDist |

Different attacks combine these two subroutines in different ways, as described below. As we will see, how an attack balances these two routines largely impacts how stealthy the attack can be made. Briefly, calls to checkAdv cannot be turned stealthy because each bad query provides just two bits of information to the attacker on average. In contrast, a call to getDist can be implemented so that a single bad query yields $\log {}^{1}/_{\eta}$ bits of information.

**An overview of existing attacks.** We briefly review how different attacks make use of checkAdv and getDist routines in the projBoundary, updateDir, and stepSize phases.

BOUNDARY ATTACK: The original decision-based attack of Brendel et al. [3] is a greedy attack. In contrast to other attacks, it only performs a heuristic, approximate projection to the model's boundary in each step.

- projBoundary: Given a misclassified point $x_b$ along the direction $\theta$ (originally a natural sample from a different class than $x$), the attack samples random points around $x_b$ and checks on which side of the boundary they fall. From this, the attack estimates a step size to project $x_b$ onto the boundary and then computes the distance dist between $x_b$ and $x$. This requires $n$ calls to checkAdv.
- updateDir: The attack is greedy and simply picks a small update direction $\delta$ at random.
- stepSize: The attack checks whether the distance to the boundary along the new direction $\theta + \delta$ is smaller than the current distance, dist. If not, the update is discarded. Note that this test can be performed with a single query to the model, with a call to checkAdv.

RAYS: This is a greedy attack similar to BOUNDARY ATTACK, tailored to the $\ell_\infty$ norm. Its search direction $\theta \in \{-1, +1\}^d$ is always a signed vector.

- projBoundary: RAYS find the current distance to the decision boundary using a binary search, by calling getDist.
- updateDir: The attack picks a new search direction by flipping the signs of a all pixels in a rectangular region of $\theta$.
- stepSize: The attack greedily checks whether the new direction improves the current distance to the boundary, by issuing a call to checkAdv. If the distance is not reduced, the update is discarded.

OPT: This attack first proposed a gradient-estimation approach to decision-based attacks.

- projBoundary: The attack starts by measuring the distance to the boundary, with a call to getDist. Specifically, it performs a binary search between $x$ and some point $\hat{x}$ of a different class along the direction $\theta$.
- updateDir: The attack estimates the gradient of the distance to the boundary along the search direction $\theta$. To this end, it samples random directions $r_1, \ldots, r_n$ and computes the distance to the boundary along $\theta + r_i$, denoted as $d_i \in \mathbb{R}^+$, for each. The estimated gradient is then:

$$\delta \leftarrow \frac{1}{n} \sum_{i=1}^{n} (\texttt{dist} - d_i) \cdot r_i \,. \tag{2}$$

  The attack uses $n$ calls to getDist to compute the boundary distance along each random direction.
- stepSize: OPT computes the step-size $\alpha$ with a *geometric search*: starting from a small step size, double it as long as this decreases the distance to the decision boundary along the new direction $\theta + \alpha \cdot \delta$. Thus, each step of the geometric search involves a call to getDist.

SIGN-OPT and HOPSKIPJUMP: These attacks are very similar and improve over OPT by using a more query-efficient gradient-estimation procedure.

- projBoundary: In HOPSKIPJUMP, this step is viewed as a boundary "projection" step which returns the point $x_b$ on the boundary, while SIGN-OPT computes the *distance* from $x$ to the boundary along $\theta$. But the two views, and their implementations, are equivalent. Both attacks use a

TABLE III: Where do decision-based attacks spend their queries? We run untargeted attacks against a ResNet-50 on ImageNet (see Section V-A for details). For each attack, we report the fraction of queries used in `checkAdv` or `getDist` routines, and the fraction of flagged queries in each routine.

| | | `checkAdv` | | `getDist` | |
|---|---|---|---|---|---|
| **Norm** | **Attack** | **all** | **fraction flagged** | **all** | **fraction flagged** |
| $\ell_2$ | BOUNDARY | 100% | 84% | 0% | – |
| | OPT | 2% | 50% | 98% | 52% |
| | SIGN-OPT | 77% | 52% | 23% | 57% |
| | HOPSKIPJUMP | 93% | 55% | 7% | 43% |
| $\ell_\infty$ | HOPSKIPJUMP | 92% | 50% | 8% | 50% |
| | RAYS | 36% | 67% | 64% | 78% |

binary search to find a point $x_b$ on the boundary, as in OPT, with a call to `getDist`.

- `updateDir`: Both attacks also sample $n$ random search directions $r_1, \ldots, r_n$. But instead of computing the distance to the boundary along each updated direction as in OPT, SIGN-OPT, and HOPSKIPJUMP simply check whether each update decreases the current distance `dist` to the decision boundary or not. The update direction is computed as

$$\delta \leftarrow \frac{1}{n} \sum_{i=1}^{n} z_i \cdot r_i \,, \tag{3}$$

  where $z_i \in \{-1, +1\}$ is one if and only if the point at distance `dist` along the direction $\theta + r_i$ is misclassified. HOPSKIPJUMP differs slightly in that the random directions $r_i$ are applied to the current point on the boundary $x_b$, and we check whether $x_b + r_i$ is misclassified or not. Compared to OPT, these attacks thus only issue $n$ calls to `checkAdv` (instead of $n$ calls to `getDist`), but the gradient estimate they compute has a higher variance.
- `stepSize`: SIGN-OPT uses the exact same geometric step-size search as OPT. HOPSKIPJUMP is slightly different from the generic algorithm described above, in that it applies the update $\delta$ to the current point on the boundary $x_b$. The attack starts from a large step size and halves it until $x_b + \alpha \cdot \delta$ is misclassified. This amounts to finding the distance to the boundary from $x_b$ along the direction $\delta$, albeit with a geometric backtracking search instead of a binary search.

Table II summarizes the calls made to `checkAdv` and `getDist` by each attack. In Table III, we show how many flagged queries and total queries are used for both routines in an untargeted attack for a standard ResNet-50 on ImageNet (where we view the class to be evaded as "flagged").

### B. Maximizing Information per Flagged Query

To design stealthy decision-based attacks, we first introduce the *entropy-per-flagged-query* metric. This is the information (measured in bits) that the attacker learns for every flagged query made to the model.

Consider an attack that calls `checkAdv`$(x, \theta + r_i, \texttt{dist}, y)$ for many random $r_i$. BOUNDARY ATTACK, HOPSKIPJUMP and SIGN-OPT do this to estimate the shape of the decision boundary. For a locally linear boundary, we expect 50% of such queries to be flagged. The attacker thus learns two bits of information per flagged query. To increase the entropy-per-flagged-query, we would need to sample the $r_i$ so that fewer `checkAdv` queries are flagged. But this requires a prior on the boundary's geometry, which is what these queries aim to learn. It thus seems hard to make this procedure stealthier.

For calls to `getDist`, a standard binary search requires $\log 1/\eta$ queries (half of which are flagged) to estimate the boundary distance up to tolerance $\eta$. A call to `getDist` thus gives $\log 1/\eta$ bits of information. So the attacker also learns an average of two bits per flagged query. However, here there is a simple way to trade-off flagged queries for non-flagged ones, which lets the attacker learn the same $\log 1/\eta$ bits of information with as little as one flagged query. All that is required is a tall building and some eggs!

**Measuring distances with one flagged query.** In the famous "egg-dropping problem", there is a building of $N$ floors, and you need to find the highest floor $n \in [1, N]$ from which an egg can be dropped without breaking. The egg breaks if and only if dropped from above some unknown floor $n$. In the simplest version of the problem, you have a single egg and must compute the value of $n$. The solution is to drop the egg from each floor consecutively starting from the first until it breaks.

We note that finding the decision boundary between $x$ and $\hat{x}$, while minimizing flagged queries, is exactly the egg-dropping problem! Assuming $\|x - \hat{x}\|_p = 1$, a search tolerance of $\eta$ yields a "building" of $N = 1/\eta$ "floors" of length $\eta$ from $\hat{x}$ to $x$. The first $n$ floors (up to the boundary) are non-flagged queries, i.e., no broken egg. All floors above $n$ are flagged queries on the wrong boundary side, i.e., a broken egg.

While a binary search minimizes the total number of queries for finding the boundary, a *line-search*—which moves from $\hat{x}$ to $x$ until the boundary is hit—is optimal for minimizing flagged queries.

Many attacks use a small search tolerance $\eta$ (on the order of $10^{-3}$), so a full line search incurs a large cost of *non-flagged* queries ($1/\eta$). This tradeoff is warranted if flagged queries are substantially more expensive than non-flagged ones, i.e., $c_{\text{flagged}} \gg c_0/\eta$. We thus consider finer-grained methods to trade

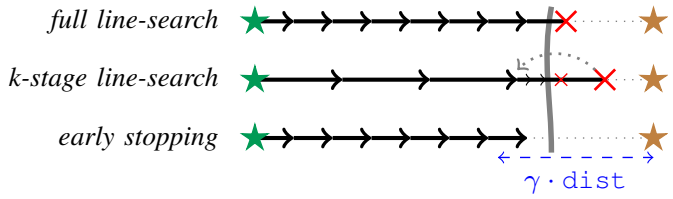



Fig. 2: Line-search strategies to find the boundary (in gray) between a benign input (green) and the original flagged input (brown). Red crosses are flagged queries.

off flagged and non-flagged queries using the general version of the egg-dropping problem.

**Trading non-flagged and flagged queries.** In the general version of the egg-dropping problem, you are given $k \geq 1$ eggs to find the safe height $n$ with a minimal number of egg drops. This problem has a standard dynamic programming solution. Asymptotically, you need $\Theta(N^{1/k})$ egg drops given $k$ eggs, as we now show for $k = 2$ eggs: first, divide the $N$ floors into $\sqrt{N}$ groups of $\sqrt{N}$ floors and do a *coarse-grained* line-search by dropping from floors $1, 1+\sqrt{N}, 1+2\sqrt{N}, \ldots$ until the first egg breaks. You now know the solution is in the previous group of $\sqrt{N}$ floors, so you do a *fine-grained* line-search in this group one floor at a time. This requires at most $2\sqrt{N}$ egg drops.

For our boundary finding problem, we can thus divide the interval between $x$ and $\hat{x}$ into $1/\eta$ intervals, and do two line searches with step-sizes respectively $\sqrt{\eta}$ and $\eta$. This will incur two flagged queries, and $2\sqrt{1/\eta}$ total queries, compared to one flagged query and $1/\eta$ total queries as above.

**A further optimization: early stopping.** Greedy attacks such as RAYS repeatedly check whether a new search direction $\theta' \leftarrow \theta + \delta$ improves upon the current adversarial distance `dist`, and only if so issue a call to `getDist` to compute the new distance $\texttt{dist}' < \texttt{dist}$. For these attacks to progress, it may not be necessary to compute $\texttt{dist}'$ *exactly*. Instead, knowing that $\texttt{dist}' \ll \texttt{dist}$ may be sufficient to know that the new direction $\theta'$ is "non-flagged" and the attack can proceed with it. We could thus stop a line search early when $\texttt{dist}' \leq \gamma \cdot \texttt{dist}$—for some $\gamma < 1$. In many cases, this lets us call `getDist` while incurring *no flagged query at all*, at the expense of a less accurate distance computation.

### C. Stealthy Variants of Decision-based Attacks

We now design stealthy variants of prior decision-based attacks, by applying the toolkit of stealthy search procedures outlined above, and illustrated in Figure 2.

**Stealthy distance computations.** The most obvious way to make existing attacks more stealthy is to instantiate every call to `getDist` with a (k-stage) line search instead of a binary search. In contrast, calls to `checkAdv` on arbitrary directions $\theta'$ are hard to make more stealthy. This change applies to the boundary distance computation in RAYS, to the gradient-estimation queries in OPT, and to the step-size searches and boundary projections in HOPSKIPJUMP, SIGN-OPT, and OPT. Since BOUNDARY ATTACK only calls `checkAdv`, it cannot easily be made more stealthy.

**Stealthy gradients.** Attacks like OPT, SIGN-OPT and HOPSKIPJUMP use most of their queries for estimating gradients. The main difference is that instead of calling `checkAdv`, OPT uses more expensive calls to `getDist` to get a better estimation. Prior work shows that this tradeoff is suboptimal in terms of total queries. However, the extra precision comes for free when we consider the cost in flagged queries! Recall that `checkAdv` yields two bits of information per flagged query, while `getDist` with a line-search yields $\log 1/\eta$ bits. Thus, OPT's gradient estimator is strictly better if we consider flagged queries. In Section IV-D we formally prove that (under mild conditions) OPT's gradient estimator gives *quadratically better convergence rates* (in terms of flagged queries) than the gradient estimators of SIGN-OPT and HOPSKIPJUMP. We can leverage this insight to design stealthy "hybrid" attacks that combine OPT's stealthy gradient estimator with efficient components of other, newer attacks. Specifically, we design a STEALTHY HSJA attack, which directly plugs in OPT's gradient estimator into HOPSKIPJUMP's otherwise more efficient attack design. (applying this change to SIGN-OPT would just yield back the OPT attack since both attacks only differ in the gradient estimator).

**Stealthy hyper-parameters.** Prior attacks were designed with the goal of minimizing the *total* number of queries. As a result, their hyper-parameters were also tuned for this metric. When considering our asymmetric query cost, existing hyper-parameters might thus no longer be optimal.

**Our attacks.** We combine the above principles to design stealthy variants of existing attacks:

- **STEALTHY RAYS**: As in the original attack, in each iteration, we first greedily check if a new search direction improves the boundary distance and then replace the binary search for the new distance by a (k-stage) line-search, optionally with early-stopping (see Section IV-B).
- **STEALTHY OPT**: The OPT attack is perfectly amenable to stealth as it only calls `getDist`. We replace the original binary search with a (k-stage) line search in each of these distance computations. When computing distances in random directions for estimating gradients, we need to select a safe starting point for the line search. If the current boundary distance is `dist`, we start the search at the point at distance $(1+\gamma) \cdot \texttt{dist}$ along $\theta'$, for $\gamma > 0$. If this point is not misclassified (i.e., the query is flagged), we return $(1 + 2 \cdot \gamma) \cdot \texttt{dist}$ as an approximate distance. If the point is misclassified (i.e., safe), we perform a line search with tolerance $1/\eta$. We use $\gamma = 1\%$ in all experiments.
- **STEALTHY HSJA**: In each iteration, we use line searches to compute the current boundary distance and the up-

dated step-size. We replace the original coarse-grained gradient estimator (which calls `checkAdv` $n$ times) with STEALTHY OPT's estimator (with $n/20$ `getDist` calls).
- **STEALTHY SIGN-OPT**: We make the same changes as for STEALTHY HSJA, except that we retain the original coarse-grained gradient estimator (otherwise this would be the same as STEALTHY OPT). To better balance the number of flagged queries used in different attack phases, we reduce the number $n$ of queries used to estimate gradients. This change is sub-optimal if we care about the attack's *total* number of queries, but is beneficial in terms of *flagged* queries as the attack now spends a larger fraction of work on queries that *can* be made more stealthy.

### D. Convergence Rates of Stealthy Attacks

Before we evaluate our stealthy attacks empirically, we show that our techniques lead to attacks that are *provably* more stealthy than prior attacks (under similar assumptions as considered in prior work).

Prior work has analyzed the convergence rate of SGD with the zero-order gradient estimation schemes used in SIGN-OPT and OPT [22, 11]. We can use these results to prove that the gradient estimation of our STEALTHY OPT attack is asymptotically more efficient (in terms of flagged queries) than the non-stealthy gradient estimation used by SIGN-OPT and HOPSKIPJUMP.

Let $g(\theta)$ be the distance to the boundary along the direction $\theta$, starting from some example $x$ (this is the function that OPT and SIGN-OPT explicitly minimize). Suppose we optimize $g$ with black-box gradient descent, using the following two gradient estimators:

- OPT: $\frac{1}{Q}\sum_{i=1}^{Q}(g(\theta+r_i)-g(\theta))\cdot r_i$ for $Q$ random Gaussian directions $r_i$.
- SIGN-OPT: $\frac{1}{Q'}\sum_{i=1}^{Q'}\texttt{sign}(g(\theta+r_i)-g(\theta))\cdot r_i$ for $Q'$ random Gaussian directions $r_i$.

We can then show the following results:

**Theorem 1** (Adapted from Liu et al. [22] (Theorem 2))**.** *Assume $g$ has gradients that are $L$-Lipschitz and bounded by $C$ (assume $L$ and $C$ are constants for simplicity). Let $d$ be the data dimensionality. Optimizing $g$ with $T$ iterations of gradient descent, using* OPT*'s gradient estimator, yields a convergence rate of $\mathbb{E}[\|\nabla g(x)\|_2^2] = \mathcal{O}(d/T)$, with $\mathcal{O}(T^2/d)$ flagged queries.*

**Theorem 2** (Adapted from Cheng et al. [11] (Theorem 3.1))**.** *Assume $g$ is $L$-Lipschitz and has gradients bounded by $C$ (assume $L$ and $C$ are constants for simplicity). Let $d$ be the data dimensionality. Optimizing $g$ with $T$ iterations of gradient descent, using* SIGN-OPT*'s gradient estimator, yields a convergence rate of $\mathbb{E}[\|\nabla g(x)\|_2] = \mathcal{O}(\sqrt{d/T})$, with $\mathcal{O}(T^2 d)$ flagged queries.*

The convergence rate of OPT is thus at least as good as that of SIGN-OPT,[3] but OPT's gradient estimator with line searches requires a factor $d^2$ fewer flagged queries. The same asymptotic result as for SIGN-OPT holds for the similar estimator used by HOPSKIPJUMP.

*Proof of Theorem 1.* Liu et al. [22] show that OPT's gradient estimator yields a convergence rate of $\mathbb{E}[\|\nabla g(x)\|_2^2] = \mathcal{O}(d/T) + \mathcal{O}(1/Q)$ (see Theorem 2 in Liu et al. [22]). To balance the two convergence terms, we set $Q = T/d$. To perform $Q$ evaluations of $g(\theta+r_i)-g(\theta)$, we need $Q+1$ calls to `getDist`. Each call makes multiple queries to the model $f$, but only one flagged query if we use a line search. This yields the number of flagged queries in the theorem ($T$ iterations with $\frac{T}{d}$ flagged queries per iteration). □

*Proof of Theorem 2.* Cheng et al. [11] show that SIGN-OPT's gradient estimator yields a convergence rate of $\mathbb{E}[\|\nabla g(x)\|_2] = \mathcal{O}(\sqrt{d/T}) + \mathcal{O}(d/\sqrt{Q'})$ (see Theorem 3.1 in Cheng et al. [11]). To balance the two convergence terms, we set $Q' = Td$. To perform $Q'$ evaluations of $\texttt{sign}(g(\theta+r_i)-g(\theta))$, one call to `getDist` and $Q'$ calls to `checkAdv` are required. Each `checkAdv` call makes a single query to the model $f$, i.e., $1/2$ flagged queries on average. This yields the number of flagged queries in the theorem ($T$ iterations with $\frac{Td}{2}$ flagged queries per iteration). □

## V. EVALUATION

We evaluate our stealthy decision-based attacks on a variety of benchmarks, in order to show that our attacks can drastically reduce the number of flagged model queries compared to the original attacks.

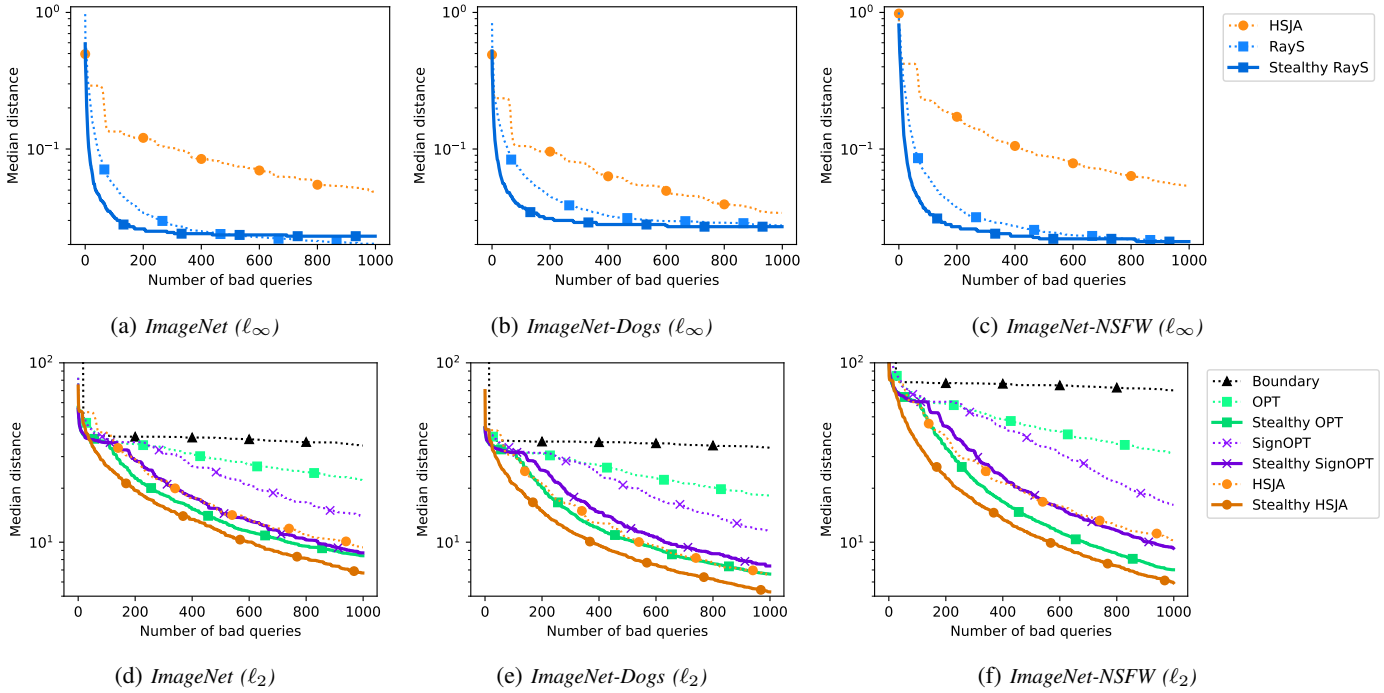
### A. Setup

**Datasets and models.** We consider four benchmarks:

- We begin with standard untargeted attacks on ImageNet against a ResNet-50 classifier. We mark a query as flagged if it is classified into the class of the original input.
- To capture more realistic security-critical scenarios, we consider a variety of binary classification tasks that aim to separate "non-flagged" from "flagged" data. As a toy benchmark, we use a binary labeling of ImageNet (hereafter ImageNet-Dogs), with all dog breeds grouped as the "flagged" class. The classifier is also a ResNet-50, with a binary head finetuned over the ImageNet training set.
- We then consider an NSFW classification task with a CLIP classifier that was used to sanitize the LAION dataset [32]. To avoid collecting a new NSFW dataset, we use a subset of ImageNet (hereafter "ImageNet-NSFW") that this classifier labels as NSFW with high confidence.[4]

[3] Note that Cheng et al. [11] provide a bound on the gradient norm, while Liu et al. [22] provide a bound on the *squared* gradient norm. Applying Jensen's inequality to the result of Theorem 2, we know that for SIGN-OPT we have $\mathbb{E}[\|\nabla g(x)\|_2^2] \geq (\mathbb{E}[\|\nabla g(x)\|_2])^2 = \mathcal{O}(d/T)$.

[4] We do not collect a new NSFW dataset due to the ethical hazards that arise from curating such sensitive data. By using a subset of ImageNet—the most popular image dataset in machine learning research—we mitigate, but do not completely eliminate [26], the potential harms of constructing an NSFW dataset.




(a) *ImageNet* ($\ell_\infty$) (b) *ImageNet-Dogs* ($\ell_\infty$) (c) *ImageNet-NSFW* ($\ell_\infty$)

(d) *ImageNet* ($\ell_2$) (e) *ImageNet-Dogs* ($\ell_2$) (f) *ImageNet-NSFW* ($\ell_2$)

Fig. 3: Our stealthy attacks find small adversarial perturbations with fewer flagged queries. For each benchmark, we report the median adversarial distance as a function of flagged queries for various $\ell_2$ attacks (top) and $\ell_\infty$ attacks (bottom). Our attack variants (full lines)–designed to be stealthy–require fewer flagged queries than the original attacks (dashed lines) to reach the same adversarial distance.

- Finally, we evaluate a black-box commercial NSFW detector, using our ImageNet-NSFW dataset. The detector returns a score from 1 to 5, denoting that the input is "highly unlikely" to "highly likely" to contain adult content or nudity. We consider a query to be flagged if it gets a score of 4 or 5.

We provide more details on datasets and models in Appendix A.

**Attacks.** We evaluate HopSkipJump and RayS for $\ell_\infty$ attacks, and Boundary Attack, Opt, Sign-Opt and HopSkipJump for $\ell_2$ attacks. We adapt each attack's official code to enable the counting of flagged queries. We use each attack's default hyper-parameters, except for some optimizations by Sitawarin et al. [33] (see Appendix A).

We further evaluate our *stealthy* versions (i.e., designed to be stealthy) of Opt, Sign-Opt, HopSkipJump, and RayS. For Stealthy Opt, Stealthy Sign-Opt, and Stealthy HSJA we split search intervals into 10,000 sub-intervals, and perform either a full line search or a two-stage line search with 100 coarse-grained and fine-grained steps. For efficiency's sake, we perform two-stage line searches in all our experiments and use the results to infer the number of queries incurred by a full line search. For Stealthy Sign-Opt, we further trade-off the query budgets for computing gradients and step-sizes by reducing the attack's default number of gradient queries $n$ by a factor $k \in \{1.5, 2.0, 2.5\}$, and find the $k$ (which we call "optimal $k$") that provides the best results. For the step-size searches, we use the same line-search procedure as in Opt.

For Stealthy RayS, we replace each binary search with a line-search of step-size $\eta = 10^{-3}$ (the default binary search tolerance for RayS) and implement early-stopping with $\gamma = 0.9$.

**Metrics.** As in prior work, we report the median $\ell_p$ norm of adversarial examples after $N$ attack queries (except we only count *flagged* queries). For each task, we run the attacks on 500 samples from the corresponding test set (for ImageNet-Dogs, we only attack images of dogs). For the attacks on the commercial NSFW detector, use use 200 samples from ImageNet-NSFW.

Our motivation for counting flagged queries is to assess whether black-box attacks are viable for attacking real security systems. We thus focus on a "low" query regime: each attack can make at most 1,000 flagged queries per sample. Prior work has considered much larger query budgets, which we disregard here as such budgets are likely not viable against systems that implement any query monitoring.

### B. Results

The main results of our evaluation appear in Figure 3. We also provide a full ablation over different attack variants and optimizations in Table IV. For all benchmarks, our stealthy attacks (with 1-stage line searches) issue significantly fewer flagged queries than the corresponding original attack.

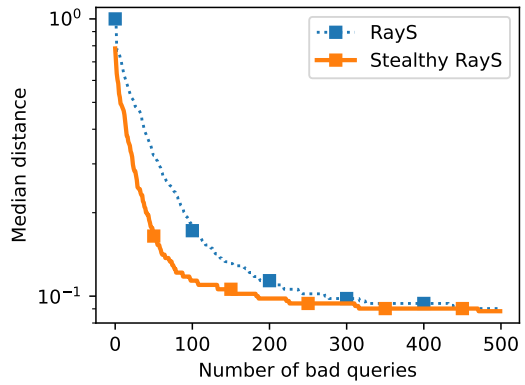



Fig. 4: Attack on a commercial black-box NSFW detector. We run RAYS and STEALTHY RAYS on 200 samples from our ImageNet-NSFW dataset. We denote a query as flagged if it is flagged as "likely" to be NSFW. The stealthy attack needs 2.2× fewer flagged queries to find adversarial perturbations of size $\epsilon = {}^{32}/_{355}$.

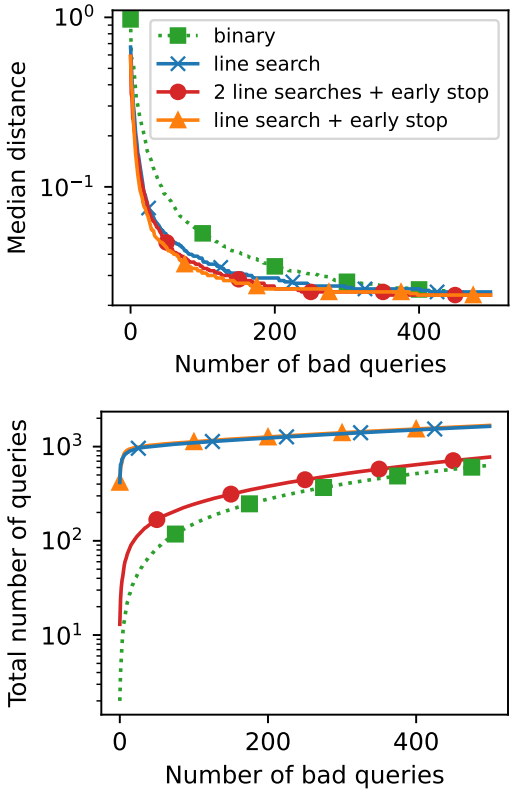



Fig. 5: Trade-offs between non-flagged and flagged queries for various search strategies in the RAYS attack on ImageNet.

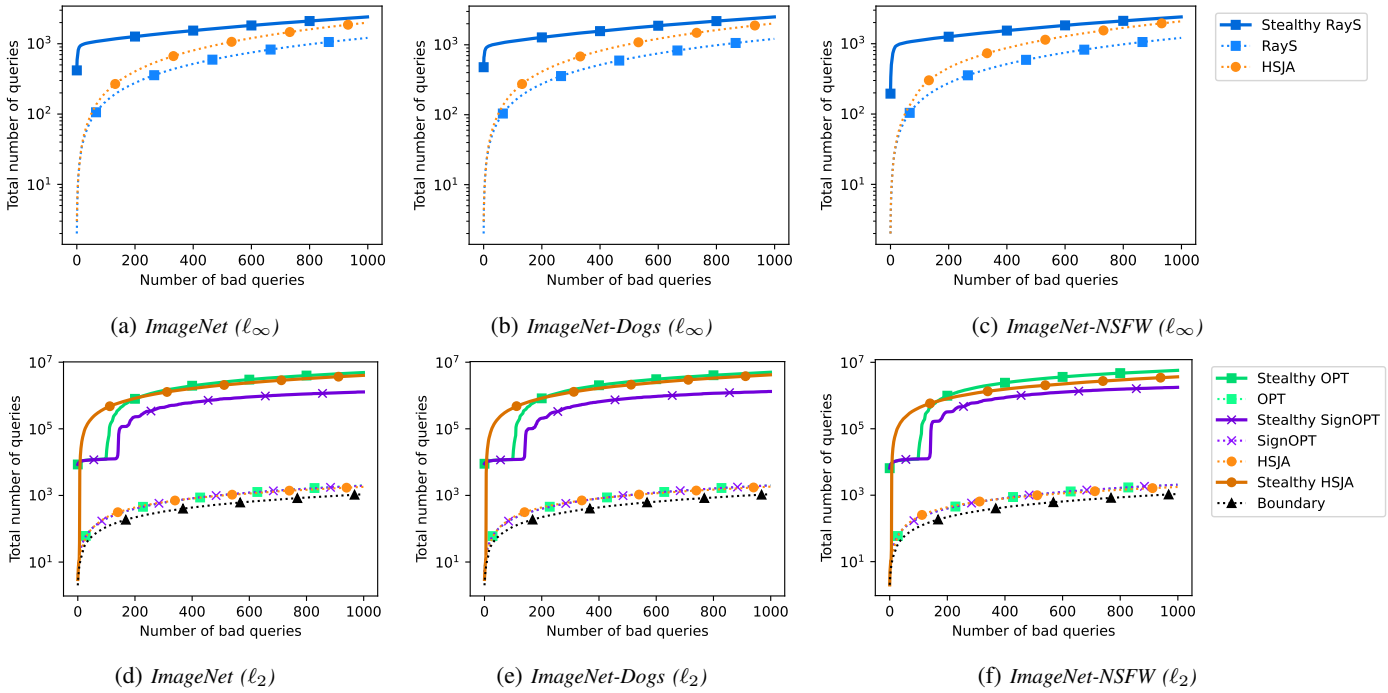
**Stealthy $\ell_\infty$ attacks are cost-effective.** Our STEALTHY RAYS attack reduces flagged queries compared to the original RAYS, which is itself more efficient than HOPSKIPJUMP. To reach a median norm of $\epsilon = {}^{8}/_{255}$, STEALTHY RAYS needs 103–181 flagged queries for the three benchmarks, 2.1–2.4× less than RAYS, and 7–17× less than HOPSKIPJUMP. As STEALTHY RAYS issues only 2.1–3.4× more queries than RAYS (see Figure 6), it is clearly cost effective if $c_{\text{flagged}} \gg c_0$.

Due to the greedy nature of RAYS, it fails to find very small perturbations. HOPSKIPJUMP thus outperforms RAYS given a large enough query budget. But this regime is likely unimportant for practical attacks, both because the query budget required is too high, and because perturbations of the size found by RAYS are likely sufficient in practice.

**Attacking a real black-box NSFW detector.** We now turn to a much more realistic attack scenario where we target a commercial black-box detector of NSFW images using STEALTHY RAYS. Few decision-based attacks in the literature have been evaluated against real black-box ML systems. In fact, many attacks require non-trivial changes to work against a real ML system which expects queries to be valid 8-bit RGB images. Notably, the binary-search tolerance $\eta$ used in the attacks we have considered is orders of magnitude smaller than the minimal distance between two RGB images. The few attacks that have been evaluated against commercial systems (e.g., the BOUNDARY ATTACK, or QEBA [20]) used a limited number of attack samples (3 to 5) due to the high query cost—and thus monetary cost—of evaluating these attacks against a commercial API. To enable a more rigorous evaluation, we focus here on RAYS—the only attack we evaluated that reliably finds small adversarial perturbations on a limited query budget (<500 queries).

Since real black-box systems expect 8-bit RGB images as input, we set RAYS's threshold $\eta$ for a binary search or line-search to ${}^{1}/_{255}$, the smallest distance between two distinct RGB images. This is much coarser than the default threshold of $\eta = 10^{-3}$, and the attack thus finds larger perturbations. Recall that in each iteration RAYS checks whether a small change to the current direction results in a smaller $\ell_\infty$ perturbation. The issue is that with discretized images, the smallest measurable change in the $\ell_\infty$ norm is ${}^{1}/_{255}$. Thus, the attack only works if small changes to the adversarial direction result in significant reductions of the $\ell_\infty$ norm. Other decision-based attacks face similar quantization issues when applied to real black-box systems. We thus encourage future work to take into account query discretization when designing black-box attacks.

We evaluate RAYS and STEALTHY RAYS on 200 images from ImageNet-NSFW. The API that we attack naturally expects to be queried with NSFW data and thus does not use any asymmetric pricing of queries (i.e., all API queries have the same cost). But we of course still distinguish flagged queries (classified as NSFW) from non-flagged ones, as such flagged queries would incur a large cost when attacking a real application that makes use of an NSFW detector model. Figure 4 shows the results. Evading this commercial detector is much harder than the prior models we attacked, presumably due to the discretization constraint described above. Our STEALTHY RAYS attack outperforms RAYS by 2.2× (we reach a median distance of ${}^{32}/_{255}$ with 79 flagged queries, while RAYS needs 172 flagged queries). These perturbations are noticeable, but




(a) *ImageNet* ($\ell_\infty$) (b) *ImageNet-Dogs* ($\ell_\infty$) (c) *ImageNet-NSFW* ($\ell_\infty$)

(d) *ImageNet* ($\ell_2$) (e) *ImageNet-Dogs* ($\ell_2$) (f) *ImageNet-NSFW* ($\ell_2$)

Fig. 6: Trade-offs between total queries and flagged queries made by different attacks. Our stealthy attacks (full lines) issue many more queries than their original counterparts (dashed lines).

preserve the images' NSFW nature. This comes at a reasonable overhead in terms of overall queries, which is 1.21× higher for STEALTHY RAYS.

**Stealthy $\ell_2$ attacks are more cost-effective than non-stealthy ones, but likely impractical.** Remarkably, while OPT is one of the earliest and least efficient decision-based attacks, our STEALTHY OPT variant is stealthier than the newer SIGN-OPT and HOPSKIPJUMP attacks. To reach a median $\ell_2$ perturbation of 10 on ImageNet, STEALTHY OPT needs 686 flagged queries, a saving of 7.3× over the original OPT, and of 1.4× compared to HOPSKIPJUMP. Our hybrid STEALTHY HSJA attack is the stealthiest attack overall. On all three benchmarks, it requires 1.47–1.82× fewer flagged queries than HOPSKIPJUMP to reach a median perturbation of 10. This shows that we can even improve the stealthiness of attacks that do not make use of many distance queries. Our techniques are thus likely also applicable to other decision-based attacks that follow HOPSKIPJUMP's blueprint.

Figure 6 shows the *total* number of queries made by our stealthy attacks. As expected, our stealthy attacks issue many more queries in total than attacks that optimize for this quantity. To reach a median perturbation of $\epsilon = 10$, our attacks make 350–1420× more total queries than the original non-stealthy attack. This large increase is only warranted if benign queries are significantly cheaper than flagged queries. This may be the case in some applications, e.g., uploading 1,000 benign images is permitted on platforms like Facebook [13], and thus likely less suspicious than a *single* flagged query. However, for less extreme asymmetries in query costs (e.g., $c_{\text{flagged}} = 10 \cdot c_0$), a less strict tradeoff between flagged and non-flagged queries is warranted. We will explore this in Section V-C.

In Figure 7, we further show the total `cost` of our attacks for various configurations of the query costs $c_0$ and $c_{\text{flagged}}$. A different attack variant is optimal depending on the cost overhead of flagged queries.

### C. Trading off Non-flagged and Flagged Queries

Our stealthy attacks in Figure 3 use full line searches, which use a *single* flagged query (and many non-flagged queries). In Figure 5 and Figure 8 we consider alternative tradeoffs. We provide a full ablation over different attack variants and optimizations in Table IV.

For $\ell_\infty$ attacks, STEALTHY RAYS with a two-stage line-search and early stopping provides a nice tradeoff: for a median perturbation of $\epsilon = 8/255$, the attack makes 1.37× more flagged queries than a full line-search, but 3.7× fewer total queries. This attack is actually *strictly better* than the original RAYS (thanks to early stopping): our attack makes 1.77× fewer flagged queries, and 8% fewer non-flagged queries!

For $\ell_2$ attacks, STEALTHY OPT with a two-stage line-search shows a nice tradeoff over the original OPT: for a median perturbation of $\epsilon = 10$, our attack makes 4× fewer flagged queries, at the expense of 5× more non-flagged queries (see Figure 8). Unfortunately, none of our stealthy attacks with two-stage line searches beat the original HOPSKIPJUMP in terms of flagged queries. Thus, attaining state-of-the-art stealthiness with our techniques does appear to come at the expense of a large overhead in non-flagged queries. As a result, improving the total cost of existing $\ell_2$ decision-based attacks may be

TABLE IV: Ablation on stealthy attack components. For attacks on ImageNet, we show the relative number of flagged queries and non-flagged queries (lower is better) compared to the original non-stealthy attack, to achieve a median $\ell_2$ perturbation of 10, or $\ell_\infty$ perturbation of $^8/_{255}$.

| Attack | Ablation | Flagged queries reduction (higher is better) | Non-flagged queries increase (lower is better) |
|---|---|---|---|
| HopSkipJump | with OPT grad estimation | 1.56× | 1418.72× |
| | with OPT grad estimation + 2-stage line search | 0.78× | 30.05× |
| Opt | with line search | 7.25× | 351.96× |
| | with 2-stage line search | 4.09× | 4.62× |
| Sign-Opt | with optimal $k$ | 1.06× | 0.93× |
| | with line search | 1.26× | 393.60× |
| | with line search + optimal $k$ | 1.81× | 386.60× |
| | with 2-stage line search + optimal $k$ | 1.77× | 4.79× |
| RayS | with line search | 1.98× | 3.42× |
| | with line search + early stopping | 2.37× | 3.42× |
| | with 2-stage line search + early stopping | 1.77× | 0.92× |

hard, and thus attacking real security-critical systems with these attacks may simply not be cost-effective.

## VI. How asymmetric are query costs in practice?

Whether the trade-offs provided by stealthy attacks are worthwhile is application-dependent. Security critical platforms such as social media websites or App stores provide few details about their filtering systems, for obvious reasons. Nevertheless, some platforms such as Facebook or Twitter (𝕏) do publish moderation policies that allow us to offer an educated guess on the relative costs of flagged and non-flagged queries in real-world systems ($c_{\text{flagged}}$ and $c_0$, respectively). We use Facebook and Twitter as case-studies below.

According to Meta, an account will "get a 1-day restriction from creating content" after seven violations of the "Community Standards", and a "a 30-day restriction from creating content" after ten violations. Stronger measures, up to account termination, are taken for further violations [14]. At the same time, users can upload up to $1{,}000$ photos per album [13], suggesting that thousands of non-flagged pictures can be uploaded without raising suspicion. Twitter's policies suggest similar numbers: users can create up to $2{,}400$ posts per day [38]; accounts get suspended after "repeated violations" (i.e., flagged queries) [37], which we speculate to mean 5–10 violations as in the case of Facebook. Given these numbers, we posit that an adversary could make at least three to four orders of magnitude more non-flagged queries than flagged queries with a single account. Under the assumption that the cost of setting up new accounts is the predominant attacker cost—and that a successful attack should not take more than a few days to run—we conclude that $c_0/c_{\text{flagged}}$ is on the order of $10^{-3}$ or $10^{-4}$.

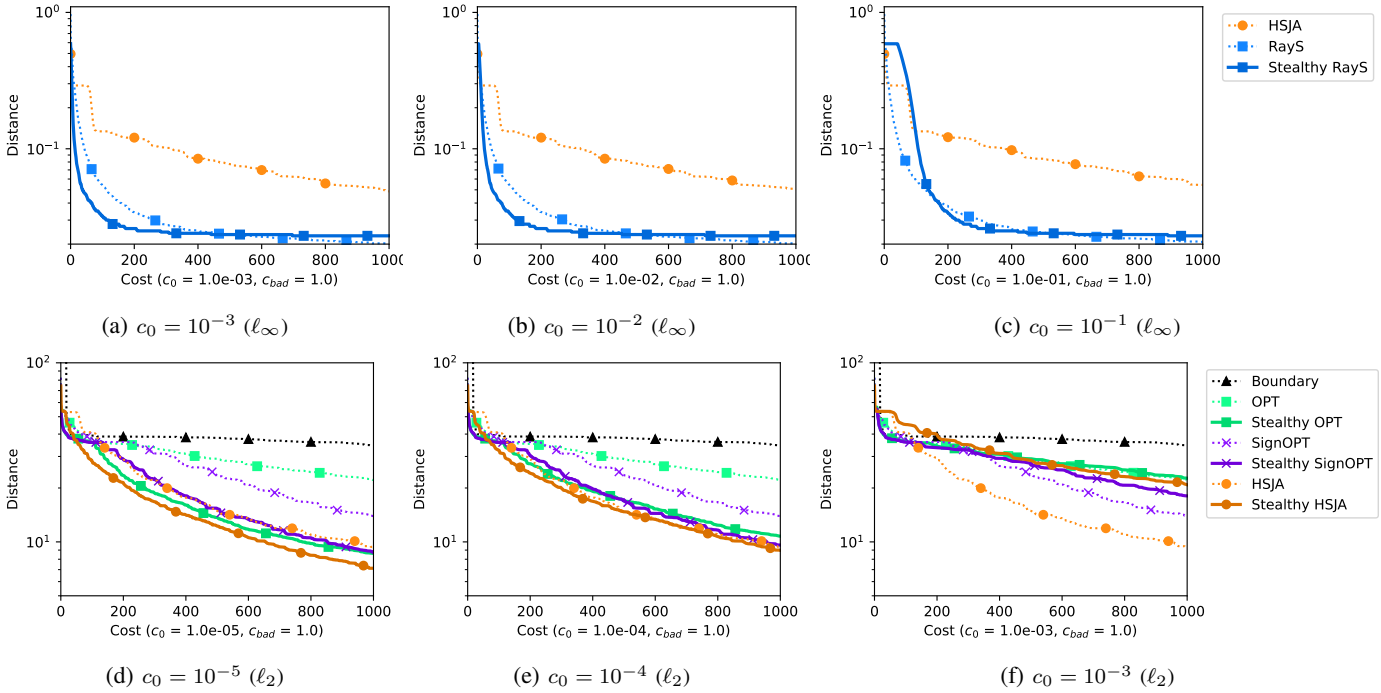
In this regime, our $\ell_\infty$ Stealthy RayS attack is clearly more cost efficient than prior $\ell_\infty$ attacks (as we can see from Figure 7a). For $\ell_2$ attacks the situation is less clear. It is possible that our Stealthy HSJA attack would be cost effective (over prior attacks) in some settings, but further attack improvements are likely necessary to obtain a stealthy attack that could realistically be applied against a deployed system.

To illustrate the practicality of the tradeoff offered by our Stealthy RayS attack, we go back to our experiment on attacking a commercial NSFW detection API from Section V-B. Suppose the attacker wants to evade detection with a perturbation of at most $\epsilon = {}^{32}/_{355}$. Our Stealthy RayS attack needs 311 overall queries, of which 79 are flagged, while RayS needs 225 queries, of which 172 are flagged. For a platform that uses a moderation policy similar to Meta's, a stealthy attacker would require setting up $^{79}/_7 \simeq 12$ accounts. In contrast, the non-stealthy attacker would need $^{172}/_7 \simeq 25$ accounts, i.e., $2.1\times$ more. Note that in this regime, neither attacker ever hits the moderation limits for unflagged queries, and thus the attack cost is purely a function of the number of flagged queries (which our stealthy attacks explicitly minimize).

## VII. Related Work

**Threat models for ML evasion attacks.** Modeling realistic ML evasion attacks is challenging [16, 1]. Our work contributes to this goal by introducing the more realistic *asymmetric query cost* metric and evaluating the feasibility of stealthy decision-based attacks. Prior work has attacked real security-critical ML systems such as malware detectors [2], copyright systems [31], or online content blockers [35, 41]. These works either assume white-box model access, or use black-box *transfer* attacks. Decision-based attacks against real ML systems have been mounted against systems that expose an explicit query API (and which do not appear to monitor queries for inappropriate data) [3, 20, 42].

**Transfer-based evasion attacks.** Transfer-based evasion attacks leverage the fact that an input that is an adversarial example for a "surrogate" white-box model could also be a successful adversarial example for another, black-box model [24]. They are perfectly stealthy (they make no flagged queries to




(a) $c_0 = 10^{-3}$ ($\ell_\infty$) (b) $c_0 = 10^{-2}$ ($\ell_\infty$) (c) $c_0 = 10^{-1}$ ($\ell_\infty$)

(d) $c_0 = 10^{-5}$ ($\ell_2$) (e) $c_0 = 10^{-4}$ ($\ell_2$) (f) $c_0 = 10^{-3}$ ($\ell_2$)

Fig. 7: Costs trade-offs of various decision-based attacks on ImageNet, for different asymmetric costs of non-flagged and flagged queries. We show how the attack cost varies for different values of the base query cost $c_0$, at a fixed cost $c_{\text{flagged}} = 1$ for bad queries. The advantage given by the stealthy attacks is reduced when the relative cost of good queries increases.

the target black-box model) but have limited success rates, as they are not model-specific, unlike decision-based attacks. The fact that they have limited success rate does not affect the performance of the attacks under our metric of counting flagged queries: transfer-based attacks make no queries at all. Moreover, previous work also explored the possibility of combining transfer with decision-based attacks by using transfer-based priors [4]. Our metric is still useful for the decision-based component of these attacks.

A further possibility to leverage transferability could be to train (or fine-tune) a surrogate model based on the outputs of the black-box model targeted by the adversary. However, to achieve this, the adversary needs to label enough training samples (both flagged and non-flagged) with the target model, to train the surrogate model. To achieve this, they would need to query the model with several flagged samples. This presents the same cost-asymmetry issue as existing decision-based attacks. While we believe that studying the amount of flagged queries needed to train such a model is out of the scope of this work, our metric would still be useful when performing this kind of evaluation.

**Detecting decision-based attacks.** Chen et al. [9] and Li et al. [21] detect decision-based attacks by monitoring sequences of user queries. Our notion of stealthiness does not consider such defenses (but it could be expanded in this way if such defenses were adopted). We aim to evade a more fundamental form of monitoring that any security-critical system likely uses: flagging and banning users who issue many "flagged" queries.

**Stealthy score-based attacks.** *Score-based attacks*, which query a model's confidence scores [8], also issue many flagged queries. Designing stealthy score-based attacks is similar to the problem of "safe black-box optimization" in reinforcement learning [39]. It is unclear whether any security-critical ML system would return confidence scores. In such a case, existing score-based attacks would be blatantly *non-stealthy*, as they typically perform zeroth-order gradient ascent starting from the original input, and thus issue *only* unsafe queries.

## VIII. Conclusion

Our paper initiates the study of *stealthy* decision-based attacks, which minimize costly *flagged* queries that are flagged by an ML system. Our "first-order" exploration of the design space for stealthy attacks shows how to equip existing attacks with stealthy search procedures, at a cost of a larger number of benign queries. Decision-based attacks may be made even stealthier by designing them *from scratch* with stealth as a primary criterion. We leave this as an open problem we hope future work can address.

We hope our paper will pave the way towards more refined analyses of the cost of evasion attacks against real ML systems. In particular, our paper suggests a new possible defense metric for defenses designed to resist black-box attacks: the number of flagged queries before an attack is effective.

## References

[1] Giovanni Apruzzese, Hyrum S Anderson, Savino Dambra, David Freeman, Fabio Pierazzi, and Kevin A Roundy.

"Real attackers don't compute gradients": Bridging the gap between adversarial ML research and practice. *arXiv preprint arXiv:2212.14315*, 2022.

[2] Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Šrndić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. Evasion attacks against machine learning at test time. In *European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 387–402. Springer, 2013.

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

[4] T. Brunner, F. Diehl, M. Le, and A. Knoll. Guessing smart: Biased sampling for efficient black-box adversarial attacks. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 4957–4965, Los Alamitos, CA, USA, nov 2019. IEEE Computer Society. doi: 10.1109/ICCV.2019.00506. URL https://doi.ieeecomputersociety.org/10.1109/ICCV.2019.00506.

[5] Business Matters. The market of Facebook accounts for sale. https://bmmagazine.co.uk/business/the-market-of-facebook-accounts-for-sale/, Aug 2020.

[6] Jianbo Chen, Michael I Jordan, and Martin J Wainwright. HopSkipJumpAttack: A query-efficient decision-based attack. In *2020 IEEE Symposium on Security and Privacy (S&P)*, pages 1277–1294. IEEE, 2020.

[7] Jinghui Chen and Quanquan Gu. RayS: A ray searching method for hard-label adversarial attack. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1739–1747, 2020.

[8] Pin-Yu Chen, Huan Zhang, Yash Sharma, Jinfeng Yi, and Cho-Jui Hsieh. Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In *Proceedings of the 10th ACM workshop on artificial intelligence and security*, pages 15–26, 2017.

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

[10] Minhao Cheng, Thong Le, Pin-Yu Chen, Jinfeng Yi, Huan Zhang, and Cho-Jui Hsieh. Query-efficient hard-label black-box attack: An optimization-based approach. *arXiv preprint arXiv:1807.04457*, 2018.

[11] Minhao Cheng, Simranjit Singh, Patrick Chen, Pin-Yu Chen, Sijia Liu, and Cho-Jui Hsieh. Sign-OPT: A query-efficient hard-label adversarial attack. *arXiv preprint arXiv:1909.10773*, 2019.

[12] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[13] Facebook. How do I add to an existing album on Facebook?, 2023. URL https://www.facebook.com/help/214757948549570. Accessed: 2023-10-12.

[14] Facebook. Restricting accounts, 2023. URL https://transparency.fb.com/en-gb/enforcement/taking-action/restricting-accounts/. Accessed: 2023-10-12.

[15] Qi-An Fu, Yinpeng Dong, Hang Su, Jun Zhu, and Chao Zhang. AutoDA: Automated decision-based iterative adversarial attacks. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 3557–3574, 2022.

[16] Justin Gilmer, Ryan P Adams, Ian Goodfellow, David Andersen, and George E Dahl. Motivating the rules of the game for adversarial example research. *arXiv preprint arXiv:1807.06732*, 2018.

[17] Google. Nudity & sexual content policy. https://support.google.com/youtube/answer/2802002?hl=en. Accessed: 2023-10-10.

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

[19] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

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

[21] Huiying Li, Shawn Shan, Emily Wenger, Jiayun Zhang, Haitao Zheng, and Ben Y Zhao. Blacklight: Scalable defense for neural networks against query-based black-box attacks. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 2117–2134, 2022.

[22] Sijia Liu, Bhavya Kailkhura, Pin-Yu Chen, Paishun Ting, Shiyu Chang, and Lisa Amini. Zeroth-order stochastic variance reduction for nonconvex optimization. *Advances in Neural Information Processing Systems*, 31, 2018.

[23] Meta. How does facebook use artificial intelligence to moderate content? https://www.facebook.com/help/1584908458516247. Accessed: 2023-10-06.

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

[25] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-

library.pdf.

[26] VU Prabhu and A Birhane. Large datasets: A pyrrhic win for computer vision. *arXiv preprint arXiv:2006.16923*, 3, 2020.

[27] Jonathan Prokos, Neil Fendley, Matthew Green, Roei Schuster, Eran Tromer, and Yinzhi Cao. Squint hard enough: Attacking perceptual hashing with adversarial machine learning. In *USENIX Security Symposium*, 2023.

[28] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pages 8748–8763. PMLR, 2021.

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

[30] Ishai Rosenberg, Asaf Shabtai, Yuval Elovici, and Lior Rokach. Adversarial machine learning attacks and defense methods in the cyber security domain. *ACM Computing Surveys (CSUR)*, 54(5):1–36, 2021.

[31] Parsa Saadatpanah, Ali Shafahi, and Tom Goldstein. Adversarial attacks on copyright detection systems. In *International Conference on Machine Learning*, pages 8307–8315. PMLR, 2020.

[32] Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, et al. LAION-5B: An open large-scale dataset for training next generation image-text models. *arXiv preprint arXiv:2210.08402*, 2022.

[33] Chawin Sitawarin, Florian Tramèr, and Nicholas Carlini. Preprocessors matter! realistic decision-based attacks on machine learning systems. *arXiv preprint arXiv:2210.03297*, 2022.

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

[35] Florian Tramèr, Pascal Dupré, Gili Rusak, Giancarlo Pellegrino, and Dan Boneh. Adversarial: Perceptual ad blocking meets adversarial machine learning. In *ACM SIGSAC Conference on Computer and Communications Security*, 2019.

[36] Florian Tramer, Nicholas Carlini, Wieland Brendel, and Aleksander Madry. On adaptive attacks to adversarial example defenses. *Advances in Neural Information Processing Systems*, 33:1633–1645, 2020.

[37] Twitter. Our range of enforcement options, 2023. URL https://help.twitter.com/en/rules-and-policies/enforcement-options. Accessed: 2023-10-12.

[38] Twitter. About x limits, 2023. URL https://help.twitter.com/en/rules-and-policies/x-limits. Accessed: 2023-10-12.

[39] Ilnura Usmanova, Andreas Krause, and Maryam Kamgarpour. Safe non-smooth black-box optimization with application to policy search. In *Learning for Dynamics and Control*, pages 980–989. PMLR, 2020.

[40] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Perric Cistac, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-Art Natural Language Processing. pages 38–45. Association for Computational Linguistics, 10 2020. URL https://www.aclweb.org/anthology/2020.emnlp-demos.6.

[41] Kan Yuan, Di Tang, Xiaojing Liao, XiaoFeng Wang, Xuan Feng, Yi Chen, Menghan Sun, Haoran Lu, and Kehuan Zhang. Stealthy porn: Understanding real-world adversarial images for illicit online promotion. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 952–966. IEEE, 2019.

[42] Baolin Zheng, Peipei Jiang, Qian Wang, Qi Li, Chao Shen, Cong Wang, Yunjie Ge, Qingyang Teng, and Shenyi Zhang. Black-box adversarial attacks on commercial speech platforms with minimal information. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 86–107, 2021.

## APPENDIX A
## ADDITIONAL EXPERIMENTAL DETAILS

### A. Datasets and Models

**ImageNet.** We run the attacks against a ResNet-50 [18] classifier trained on ImageNet [12]. We use the model weights provided as part of the torchvision library [25], which reach $76.13\%$ validation accuracy. When running the attacks, we use ImageNet's validation set and we skip the samples that are already classified incorrectly by the model.

**ImageNet-Dogs.** We create a binary classification task from ImageNet by considering as "flagged" the images belonging to classes of dog breeds (i.e., the classes with indices included in the range $[151, 268]$) and as "non-flagged" the images belonging to all the other classes. We create training and validation sets in this way from the respective splits of ImageNet. Then, we take the ResNet-50 provided by torchvision, change the last linear layer to a layer with one output, and fine-tune this model for one epoch on the training set, using Adam [19] with learning rate $10^{-3}$. Training the model takes around 1 hour using an Nvidia RTX A6000. The final model has $96.96\%$ accuracy, $87.14\%$ precision, and $87.10\%$ recall on the validation set. Since we are interested in creating adversarial examples for the "flagged" images, we only attack the images in the validation set that are correctly classified as "flagged" (i.e., as dogs) by the fine-tuned model.

**ImageNet-NSFW.** As mentioned in Section V-B, we also evaluate the attacks on the NSFW content detector shared by Schuhmann et al. [32]. This classifier takes as input CLIP [28] embeddings of images and outputs a confidence in $[0, 1]$. We use the CLIP implementation provided by the HuggingFace Transformers library [40] to extract the CLIP embeddings from the input images. To create an evaluation set of NSFW images, we select the subset of 1,000 images in the ImageNet validation set that the NSFW content detector classifies as NSFW with highest confidence (it is well known that ImageNet contains NSFW content [26]). When attacking the model, we consider an attack to be successful if the confidence of the detector drops below $0.5$.

### B. Attack Hyper-parameters

**BOUNDARY ATTACK.** We use the official implementation[5], which is part of from Foolbox [29], with default hyper-parameters on all tasks.

**HOPSKIPJUMP.** We use the official implementation.[6] Following Sitawarin et al. [33], we set $\texttt{gamma} = 10{,}000$ (this hyper-parameter is used to determine the binary search threshold), as this gives better results.

**RAYS and STEALTHY RAYS.** We use the official implementation.[7] The attack has no hyper-parameters. The default binary search tolerance is $\eta = 10^{-3}$. For the line-search in STEALTHY RAYS we use the same step size of $10^{-3}$ and perform either a full line-search or a two-stage search by first dividing the $N$ search intervals into coarse groups of size $\sqrt{N}$. For attacking the commercial black-box NSFW classifier in Section V-B, we set the binary search tolerance and line-search step-size to $\eta = 1/255$ and perform a full line-search. For the early-stopping optimization, we end a line search if $\texttt{dist}' < 0.9 \cdot \texttt{dist}$.

In Figure 3, Figure 4 and Figure 6, the STEALTHY RAYS attack is the version with a full line-search and early-stopping.

**OPT and STEALTHY OPT.** We use the official implementation.[8] Following Sitawarin et al. [33], we set $\beta = 10^{-2}$ (this hyper-parameter is used to determine the binary search threshold).

For STEALTHY OPT, we do line searches for gradient estimation in the interval $[0.99 \cdot \texttt{dist}, 1.01 \cdot \texttt{dist}]$, where $\texttt{dist}$ is the current adversarial distance. For computing step sizes, we do a line search in the interval $[0.99 \cdot \texttt{dist}, \texttt{dist}]$, since we only care about the new distance if it improves upon the current one. We split this interval into $N = 10{,}000$ sub-intervals and perform a 2-stage line-search with 100 coarse-grained steps and 100 fine-grained steps. For efficiency sake, we *batch* the line-search by calling the model on two batches of size 100, one for all coarse-grained steps, and one for all fine-grained steps. To count the number of flagged queries and total queries, we assume that the line-search queries were performed one-by-one. If the first query in a line search is not safe (i.e., the boundary distance is larger than $1.01 \cdot \texttt{dist}$, we approximate the distance by $\texttt{dist}' \approx 2 \cdot \texttt{dist}$.

In Figure 3 and Figure 6, the STEALTHY OPT attack is the version with a full line search.

**SIGN-OPT and STEALTHY SIGN-OPT.** We use the official implementation.[9] Following Sitawarin et al. [33], we set $\beta = 10^{-2}$ (this hyper-parameter is used to determine the binary search threshold).

For STEALTHY SIGN-OPT, we do the same line search procedure as STEALTHY OPT for computing step sizes. We change the default number of gradient estimation queries per iteration from $n = 200$ to $n/k$ for $k \in \{1.5, 2, 2.5, 3\}$, i.e., $n \in \{67, 80, 100, 133\}$.

In Figure 3 and Figure 6, the STEALTHY SIGN-OPT attack uses a full line-search, and $k = 2.5$.

### C. Compute and code

We run every attack on one Nvidia RTX 3090, and the time to run the attacks on 500 samples ranges from twelve hours, for the attacks ran with binary search, to more than three days for the slowest attacks (e.g. OPT) ran with

[5]https://github.com/bethgelab/foolbox/blob/1c55ee/foolbox/attacks/boundary_attack.py

[6]https://github.com/Jianbo-Lab/HSJA/blob/daecd5/hsja.py

[7]https://github.com/uclaml/RayS/blob/29bc17/RayS.py

[8]https://github.com/cmhcbb/attackbox/blob/65a82f/attack/OPT_attack.py

[9]https://github.com/cmhcbb/attackbox/blob/65a82f/attack/Sign_OPT.py

line search. We wrap all the attack implementations in a common set-up for which we use PyTorch [25]. The code can be found at the following URL: https://anonymous.4open.science/r/realistic-adv-examples-CD4C/. The checkpoints of the model we trained, the NSFW classifier we ported from Keras to PyTorch, and the outputs of this model on the ImageNet train and validation datasets can be found at the following URL: https://osf.io/bhfcj/files/osfstorage?view_only=b55a3077521242a287ba957bd461fe59.

## APPENDIX B
## ADDITIONAL FIGURES

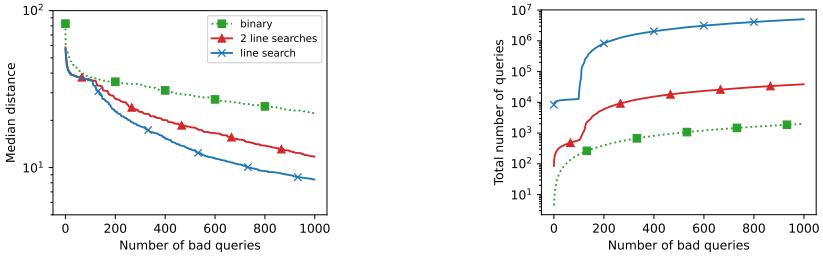



Fig. 8: Trade-offs between non-flagged and flagged queries for different search strategies in the STEALTHY OPT attack. A full line search makes one flagged query and up to 10,000 non-flagged queries. The version with two searches makes two flagged queries and up to $2 \cdot 100$ non-flagged queries.

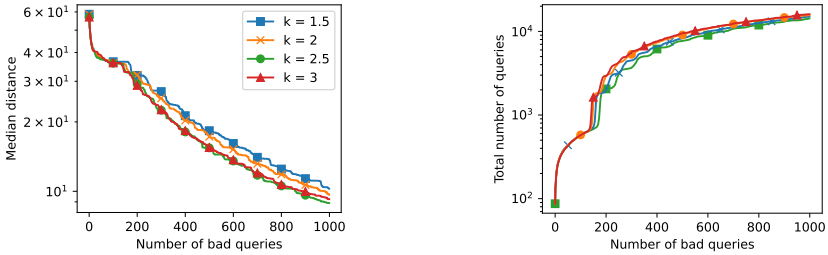



Fig. 9: Influence of the hyper-parameter $k$ in the STEALTHY SIGN-OPT attack (the reduction in the number of gradient estimation queries per iteration). The best results are obtained with $k = 2.5$.

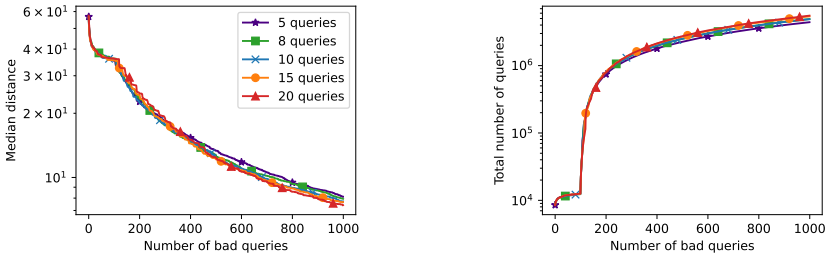



Fig. 10: Influence of the number of directions computed for the gradient estimation in the STEALTHY OPT attack. The best results are obtained with $q = 10$, which is the original value from Cheng et al. [10].

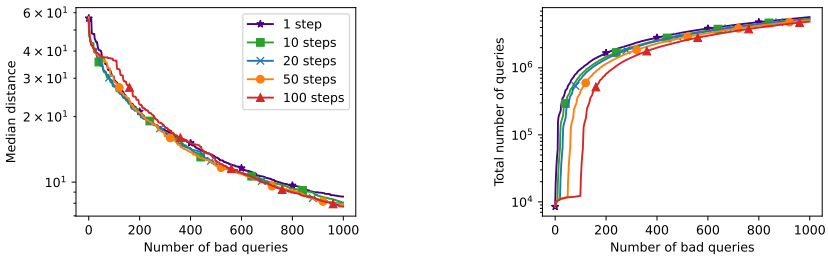



Fig. 11: Influence of the number of directions tested for the initialization in the STEALTHY OPT attack. The best results are obtained with $n = 100$, when considering a larger number of queries, even though the difference between the different values is small.