Evading Black-box Classifiers Without Breaking Eggs

Edoardo Debenedetti [1] **Nicholas Carlini** [2] **Florian Tramèr** [1]

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 flawed. Most 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*: queries detected as "bad" 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 "bad" queries, which likely renders them ineffective against security-critical systems. We then design new attacks that reduce the number of bad queries by 1.5–7.3×, but often at a significant increase in total (non-bad) queries. We thus pose it as an open problem to build black-box attacks that are more effective under realistic cost metrics.

1. Introduction

Adversarial examples (Szegedy et al., 2014; Biggio et al., 2013) 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 (Tramèr et al., 2019; Prokos et al., 2023; Rosenberg et al., 2021). In such security-critical uses of ML, the attacker often only has *black-box* access to the ML model's decisions.

Decision-based attacks (Brendel et al., 2017) 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. (2017) required over 10^5 model queries to reliably find small adversarial perturbations. Subsequent work (Cheng et al., 2018; 2019; Chen et al., 2020a; Chen & Gu, 2020; Li et al., 2020) optimized for this metric of "total number of model queries", and reduced it by 1–3 orders of magnitude.

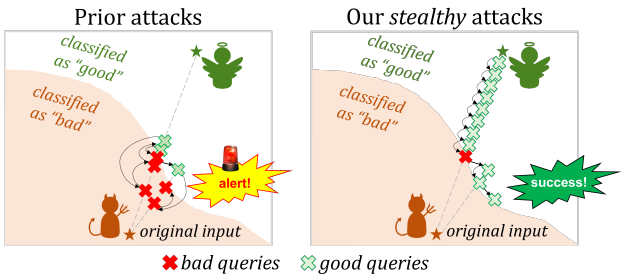


Figure 1. Existing decision-based attacks (left) make many "bad" queries, that get classified into the class that the attacker aims to evade. In security-critical applications, such bad queries likely trigger additional security mechanisms, thus raising the cost of the attack. Our *stealthy* attacks (right) trade-off bad queries for good ones, to find adversarial examples without raising security alerts.

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 flagged by the system (e.g., offensive content) triggers additional security measures that carry a high cost for the attacker—up to account termination. Thus, we argue that black-box attacks should strive to be *stealthy*, by minimizing the number of "bad" queries flagged by the ML system.

We find that existing attacks are not stealthy: over 50% of the queries they make are bad. We then show how to reduce the number of bad queries for a class of attacks that measure distances to the model's boundary along random directions (e.g., OPT (Cheng et al., 2018), SIGN-OPT (Cheng et al., 2019) and RAYS (Chen & Gu, 2020)). Inspired by the famous "egg-dropping problem" (Alves et al.), we modify these attacks to trade-off bad queries for benign ones.

We evaluate our attacks on three classification tasks: ImageNet, dog vs. not-dog in ImageNet, and NSFW content (Schuhmann et al., 2022). Our stealthy attacks reduce the number of bad queries of the original attacks by 1.5–7.3×. Notably, on ImageNet, our stealthy variant of the OPT attack outperforms SIGN-OPT and HOPSKIPJUMP in terms

[1]Department of Computer Science, ETH Zürich, Switzerland [2]Google. Correspondence to: Edoardo Debenedetti <edebenedetti@inf.ethz.ch>.

2^{nd} AdvML Frontiers workshop at 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.

of *bad* queries, despite the two latter attacks issuing fewer queries *in total*. Yet, our most stealthy ℓ_2 attacks incur a large increase in good queries (350–1,400×). The tradeoff is better for ℓ_∞ attacks: our stealthy variant of the RAYS attack reduces bad queries by 2.1–2.5× over RAYS and 6–17× over HOPSKIPJUMP, while making 2.1–3.4× more benign queries than RAYS. We use the stealthy RAYS attack to evade a commercial black-box NSFW image detector, with 2.2× fewer bad queries than the original RAYS attack.

Overall, our results suggest that many decision-based attacks are far from stealthy, and that stealthier attacks are often only viable if the cost of bad queries far outweighs that of good queries (especially for ℓ_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.

2. 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 ℓ_p norm and threshold ϵ.

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.

We discuss related work about decision based attacks, their detection, and stealthy attacks in Appendix A.

3. 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).

But we argue that query costs are rarely uniform in practical security-critical systems. This is because, in such systems, the goal of a ML model is usually to detect "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 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.

As an example, consider an attacker who tries to upload inappropriate content to a social media website. Every uploaded image passes through a ML model that flags inappropriate content. Benign content is very rarely flagged and thus carries little cost. But if a query *is* flagged as inappropriate, the system blocks the contents and may take further costly actions (e.g., account throttling or suspension).

We now formalize this asymmetry. Assume one or more of the classifier's output classes $\mathcal{Y}$ are "bad", denoted as $\mathcal{Y}_{\text{bad}}$. The attacker is given an input (x, y) where $y \in \mathcal{Y}_{\text{bad}}$ (e.g., x is a NSFW image) and wants to find an adversarial example $\hat{x}$ where $f(\hat{x}) \notin \mathcal{Y}_{\text{bad}}$. All queries to the model f carry a base cost c_0, due to data processing, network bandwidth, or 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 (Facebook, 2023). However, for queries x that are flagged as inappropriate (i.e., $f(x) \in \mathcal{Y}_{\text{bad}}$), the cost c_{bad} 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 (Business Matters, 2020)), this places a significantly higher cost on queries flagged as bad, i.e., $c_{\text{bad}} \gg c_0$.

We thus argue that decision-based attacks should strive to minimize the cost defined as: $\texttt{cost} \coloneqq Q_{\text{total}} \cdot c_0 + Q_{\text{bad}} \cdot c_{\text{bad}}$, where Q_{bad} is the number of bad model queries ($f(x) \in \mathcal{Y}_{\text{bad}}$), Q_{total} is the total number of queries—including bad ones—and $c_{\text{bad}} \gg c_0$. We call attacks that minimize this asymmetric cost *stealthy*.

Existing attacks are not stealthy. No existing black-box attack considers such asymmetric query costs. As a result, these attacks issue a large number of bad queries. We illustrate this with an untargeted attack on ImageNet.[1] In Table 1 in the appendix, we show the number of total queries Q_{total} and bad queries Q_{bad} made by various ℓ_2 and ℓ_∞ decision-based attacks on a ResNet-50 classifier. In all cases, half or more of the attacker's queries are "bad" (i.e., they get the class label that was to be evaded). Despite differences in the fraction of bad queries for each attack, attacks that make fewer total queries also make fewer bad queries. But this begs the question of whether we could design attacks that issue far fewer bad queries in total. The remainder of this paper answers this question.

Selecting the values of c_0 and c_{bad}. The true cost of a query (whether good or bad) 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_{total} and Q_{bad}, so that the attack cost can be calculated for

[1] 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 "bad" and all other classes as "good". That is, for an input (x, y) we set $\mathcal{Y}_{\text{bad}} = \{y\}$ 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.

any domain-specific values of c_0 and c_{bad}.

In this paper, we often make the simplifying assumption that $c_0 = 0, c_{\text{bad}} = 1$, a special case that approximates the attack cost when $c_{\text{bad}} \gg c_0$. In this special case, the attacker solely aims to minimize bad queries, possibly at the expense of a large increase in total queries. We will however also consider less extreme trade-offs between these two values.

4. Designing Stealthy Decision-based Attacks

To begin, we explore the design space of stealthy decision-based attacks, which minimize the total number of bad 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 1 in the appendix, 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* bad queries for good ones.

4.1. How do Decision-based Attacks Work?

Most decision-based attacks follow the same blueprint (Fu et al., 2022). For an input (x, y), the attacks first pick an *adversarial direction* $\theta \in [0,1]^d$ and find the ℓ_p distance to the model's decision boundary from x along the direction θ. They then iteratively perturb θ to minimize the boundary distance along the new direction. In each iteration, the attacks compute an update direction δ and step-size α and make an update step $\theta' \leftarrow \theta + \alpha \cdot \delta$, and then compute the new boundary distance from x along θ'.

Decision-based attacks use two fundamental subroutines:

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

Different attacks combine these two subroutines in different ways, as we describe in Appendix C. As we will see, how an attack balances these two routines largely impacts how stealthy the attack can be made.

4.2. Maximizing Information per Bad Query

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

Consider an attack that calls $\texttt{checkAdv}(x, \theta + r_i, \texttt{dist}, y)$ 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 bad. The attacker thus learns two bits of information per bad query. To increase the entropy-per-bad-query, we would need to sample the r_i so that fewer $\texttt{checkAdv}$ queries are bad. 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.

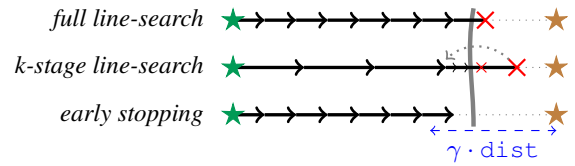


Figure 2. Line-search strategies to find the boundary (in gray) starting from a benign point (green) towards the original bad input (brown). Red crosses denote bad queries.

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

Measuring distances with one bad 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 iff 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, starting from the first, until it breaks.

We note that finding the decision boundary between x and $\hat{x}$, while minimizing bad queries, is exactly the egg-dropping problem! Assuming $\|x - \hat{x}\|_p = 1$, a search tolerance of η yields a "building" of $N = 1/\eta$ "floors" of length η from $\hat{x}$ to x. The first n floors (up to the boundary) are good queries, i.e., no broken egg. All floors above n are bad 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 bad queries. Many attacks use a small search tolerance η (on the order of 10^{-3}), so a full line-search incurs a large cost of *good* queries ($1/\eta$). We thus consider finer-grained methods to trade-off bad and good queries.

Trading good and bad 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 drops.

Asymptotically, you need $\Theta(N^{1/k})$ 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 $2\sqrt{N}$ 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 η. This will incur two bad queries, and $2\sqrt{1/\eta}$ total queries, compared to one bad 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 be unnecessary to compute $\texttt{dist}'$ *exactly*. Instead, knowing that $\texttt{dist}' \ll \texttt{dist}$ may be enough to pick the new direction θ' as it is "good". 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 bad query at all*, at the expense of a less accurate distance computation.

4.3. Stealthy Variants of Decision-based Attacks

We now design stealthy variants of prior attacks, by applying the toolkit of stealthy search procedures outlined above.

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 θ' 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 bad queries! Recall that `checkAdv` yields two bits of information per bad query, while `getDist` with a line-search yields $\log 1/\eta$ bits. Thus, OPT's gradient estimator is strictly better if we consider bad queries. In Appendix D, we formally prove that (under mild conditions) OPT's gradient estimator gives *quadratically better convergence rates* (in terms of bad queries) than the gradient estimators of SIGN-OPT and HOPSKIPJUMP. We can leverage this to design stealthy "hybrid" attacks that combine OPT's stealthy gradient estimator with efficient components of newer attacks.

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. We provide additional implementation details for all attacks in Appendix B.2.

- **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 4.2).
- **STEALTHY OPT**: The OPT attack is perfectly amenable to stealth as it only calls `getDist`. We replace the original binary search by a (k-stage) line-search in each of these distance computations.
- **STEALTHY HSJA**: In each iteration, we use line searches to compute the current boundary distance, and the update step-size. We replace the original 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 binary gradient estimator (otherwise this would be the same as STEALTHY OPT). Further, we optimize the hyper-parameters to make the attack stealthier.

5. Evaluation

We evaluate our stealthy decision-based attacks on a variety of benchmarks, to show that our attacks can greatly reduce the number of bad queries compared to the original attacks.

5.1. Setup

Datasets and models. We consider four benchmarks: untargeted attacks on ImageNet against a ResNet-50 classifier, attacks against on a binary version of ImageNet, hereafter ImageNet-Dogs, where positives are dog classes and negatives are non-dog ones, attacks against a CLIP-based NSFW detector using a subset of NSFW images in ImageNet[2], and, finally, attacks against a commercial NSFW detector, using the same dataset as the previous benchmark. We provide

[2]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 ML research—we mitigate, but do not completely eliminate (Prabhu & Birhane, 2020), the potential harms of building a NSFW dataset.

more details about the setup in Appendix B.

Attacks. We evaluate BOUNDARY ATTACK, OPT, SIGN-OPT and HOPSKIPJUMP for ℓ_2 attacks, and HOPSKIPJUMP and RAYS for ℓ_∞ attacks. We adapt each attack's official code to enable counting of bad queries. We use each attack's default hyper-parameters, except for some optimizations by Sitawarin et al. (2022) (see Appendix B).

We further evaluate our stealthy versions of OPT, SIGN-OPT, HOPSKIPJUMP and RAYS. For former three, for efficiency 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 $k \in \{1.5, 2.0, 2.5\}$. 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 use early-stopping with $\gamma = 0.9$.

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

Our motivation for counting bad 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 bad 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.

5.2. 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 4. For all benchmarks, our stealthy attacks (with 1-stage line searches) issue significantly fewer bad queries than the corresponding original attack.

ℓ_2 attacks. 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 ℓ_2 perturbation of 10 on ImageNet, STEALTHY OPT needs 686 bad queries, a saving of $7.3\times$ over the original OPT, and of $1.4\times$ compared to HOPSKIPJUMP. Our hybrid STEALTHY HSJA attack is the stealthiest attack overall. On all three benchmarks, it requires 1.47–$1.82\times$ fewer bad 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 in the appendix 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–$1420\times$ more total queries than the original non-stealthy attack. This large increase is only warranted if benign queries are significantly cheaper than bad queries. This may be the case in some applications, e.g., uploading 1,000 benign images is permitted on platforms like Facebook (Facebook, 2023), and thus likely less suspicious than a *single* bad query. However, for less extreme asymmetries in query costs (e.g., $c_{\text{bad}} = 10 \cdot c_0$), a less strict tradeoff between bad and good queries is warranted. We will explore this in Section 5.3. In Figure 11, we further show the total cost of our attacks for various configurations of the query costs c_0 and c_{bad}. A different attack variant is optimal depending on the cost overhead of bad queries.

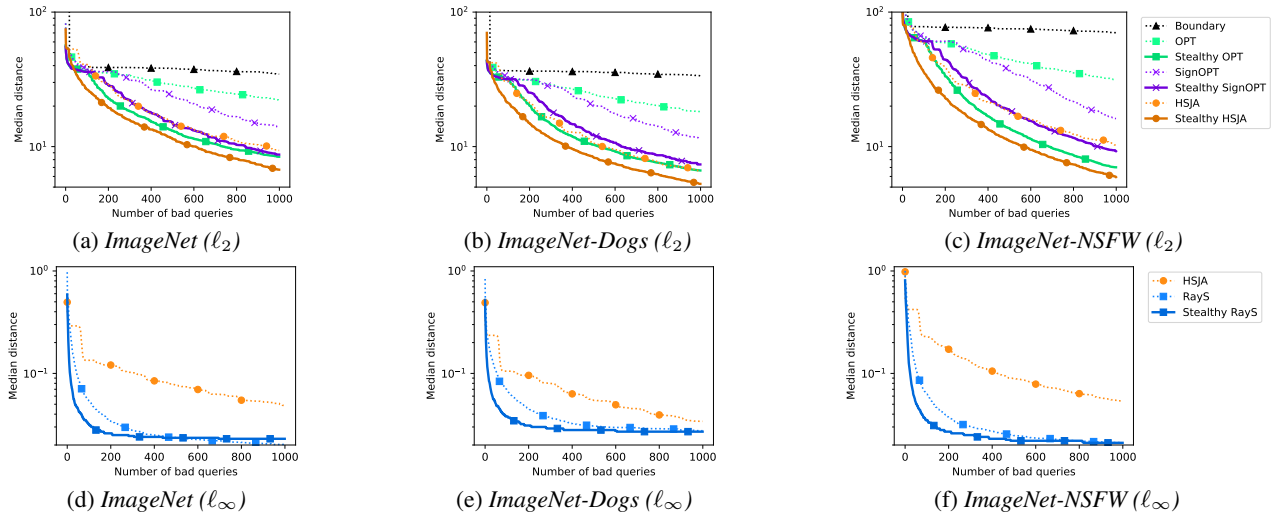
ℓ_∞ attacks. The cost-effectiveness of stealthy ℓ_∞ attacks is better. Our STEALTHY RAYS attack reduces bad queries compared to the original RAYS, which is itself more efficient than HOPSKIPJUMP. To reach a median norm of $\epsilon = 8/255$, STEALTHY RAYS needs 103–181 bad queries for the three benchmarks, 2.1–$2.4\times$ less than RAYS, and 7–$17\times$ less than HOPSKIPJUMP. As STEALTHY RAYS issues only 2.1–$3.4\times$ more queries than RAYS (see Figure 6), it is clearly cost effective if $c_{\text{bad}} \gg c_0$.

5.3. Trading off Good and Bad Queries

Our stealthy attacks in Figure 3 use full line-searches, which use a *single* bad query (and many good queries). In Figures 4 and 7 we consider alternative tradeoffs. We provide a full ablation over different attacks' optimizations in Table 4.

For ℓ_∞ 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 $1.37\times$ more bad queries than a full line-search, but $3.7\times$ fewer total queries. This attack is actually *strictly better* than the original RAYS (thanks to early stopping): our attack makes $1.77\times$ fewer bad queries, and 8% fewer good queries!

For ℓ_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 $4\times$ fewer bad queries, at the expense of $5\times$ more good queries (see Figure 7). Unfortunately, none of our stealthy attacks with two-stage line searches beat the original HOPSKIPJUMP in terms of bad queries. Thus, attaining state-of-the-art stealthiness with our techniques does appear to come at the expense of a large overhead in good queries. As a result, improving the total cost of existing ℓ_2 decision-based attacks may be hard, and thus attacking real security-critical systems with



(a) *ImageNet* (ℓ_2) (b) *ImageNet-Dogs* (ℓ_2) (c) *ImageNet-NSFW* (ℓ_2)

(d) *ImageNet* (ℓ_∞) (e) *ImageNet-Dogs* (ℓ_∞) (f) *ImageNet-NSFW* (ℓ_∞)

Figure 3. Our stealthy attacks find small adversarial perturbations with fewer bad queries. For each benchmark, we report the median adversarial distance as a function of bad queries for various ℓ_2 attacks (top) and ℓ_∞ attacks (bottom). Our stealthy attack variants (full lines) require fewer bad queries than the original attacks (dashed lines) to reach the same adversarial distance.

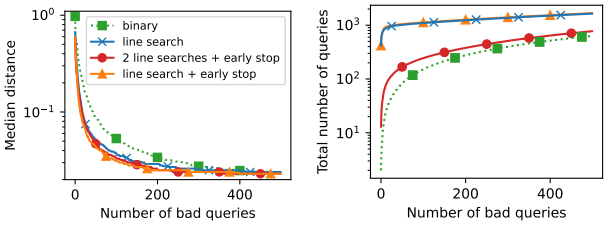


Figure 4. Trade-offs between good and bad queries for various search strategies in the RAYS attack on ImageNet.

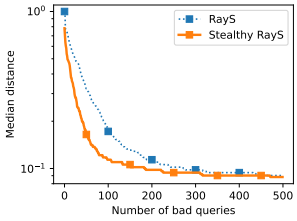


Figure 5. Attack on a commercial black-box NSFW detector. We run RAYS and STEALTHY RAYS on 200 samples from our ImageNet-NSFW dataset. We denote a query as bad if it is flagged as "likely" to be NSFW. The stealthy attack needs $2.2\times$ fewer bad queries to find adversarial perturbations of size $\epsilon = {}^{32}/_{355}$.

these attacks may simply not be cost-effective.

5.4. Attacking a 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. The few attacks that have been evaluated against commercial systems (e.g., the BOUNDARY ATTACK, or QEBA (Li et al., 2020)) 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 η 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. Other decision-based attacks face similar quantization issues when applied to real black-box systems.

Figure 5 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 $2.2\times$ (we reach a median distance of ${}^{32}/_{255}$ with 79 bad queries, while RAYS needs 172). These perturbations are noticeable, but preserve the images' NSFW nature.

6. Conclusion

Our paper initiates the study of *stealthy* decision-based attacks, which minimize costly *bad* queries that are flagged by a 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 is 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 metric for defenses designed to resist black-box attacks: the number of bad queries before an attack is effective.

References

Alves, G., Pilling, G., Innes, J., Bari, R., Quang, N., Silverman, J., Kau, A., and Khim, J. Egg dropping. https://brilliant.org/wiki/egg-dropping. Accessed: 2023-01-24.

Apruzzese, G., Anderson, H. S., Dambra, S., Freeman, D., Pierazzi, F., and Roundy, K. A. "real attackers don't compute gradients": Bridging the gap between adversarial ml research and practice. *arXiv preprint arXiv:2212.14315*, 2022.

Biggio, B., Corona, I., Maiorca, D., Nelson, B., Šrndić, N., Laskov, P., Giacinto, G., and Roli, F. Evasion attacks against machine learning at test time. In *European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 387–402. Springer, 2013.

Brendel, W., Rauber, J., and Bethge, M. Decision-based adversarial attacks: Reliable attacks against black-box machine learning models. *arXiv preprint arXiv:1712.04248*, 2017.

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

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

Chen, J., Jordan, M. I., and Wainwright, M. J. HopSkipJumpAttack: A query-efficient decision-based attack. In *2020 IEEE Symposium on Security and Privacy (S&P)*, pp. 1277–1294. IEEE, 2020a.

Chen, P.-Y., Zhang, H., Sharma, Y., Yi, J., and Hsieh, C.-J. 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*, pp. 15–26, 2017.

Chen, S., Carlini, N., and Wagner, D. Stateful detection of black-box adversarial attacks. In *Proceedings of the 1st ACM Workshop on Security and Privacy on Artificial Intelligence*, pp. 30–39, 2020b.

Cheng, M., Le, T., Chen, P.-Y., Yi, J., Zhang, H., and Hsieh, C.-J. Query-efficient hard-label black-box attack: An optimization-based approach. *arXiv preprint arXiv:1807.04457*, 2018.

Cheng, M., Singh, S., Chen, P., Chen, P.-Y., Liu, S., and Hsieh, C.-J. Sign-opt: A query-efficient hard-label adversarial attack. *arXiv preprint arXiv:1909.10773*, 2019.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Facebook. How do I add to an existing album on Facebook?, 2023. URL https://www.facebook.com/help/214757948549570.

Fu, Q.-A., Dong, Y., Su, H., Zhu, J., and Zhang, C. AutoDA: Automated decision-based iterative adversarial attacks. In *31st USENIX Security Symposium (USENIX Security 22)*, pp. 3557–3574, 2022.

Gilmer, J., Adams, R. P., Goodfellow, I., Andersen, D., and Dahl, G. E. Motivating the rules of the game for adversarial example research. *arXiv preprint arXiv:1807.06732*, 2018.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Li, H., Xu, X., Zhang, X., Yang, S., and Li, B. Qeba: Query-efficient boundary-based blackbox attack. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 1221–1230, 2020.

Li, H., Shan, S., Wenger, E., Zhang, J., Zheng, H., and Zhao, B. Y. Blacklight: Scalable defense for neural networks against {Query-Based}{Black-Box} attacks. In *31st USENIX Security Symposium (USENIX Security 22)*, pp. 2117–2134, 2022.

Liu, S., Kailkhura, B., Chen, P.-Y., Ting, P., Chang, S., and Amini, L. Zeroth-order stochastic variance reduction for nonconvex optimization. *Advances in Neural Information Processing Systems*, 31, 2018.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates, Inc., 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

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

Prokos, J., Fendley, N., Green, M., Schuster, R., Tromer, E., and Cao, Y. Squint hard enough: Attacking perceptual hashing with adversarial machine learning. In *USENIX Security Symposium*, 2023.

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pp. 8748–8763. PMLR, 2021.

Rauber, J., Brendel, W., and Bethge, M. 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.

Rosenberg, I., Shabtai, A., Elovici, Y., and Rokach, L. Adversarial machine learning attacks and defense methods in the cyber security domain. *ACM Computing Surveys (CSUR)*, 54(5):1–36, 2021.

Saadatpanah, P., Shafahi, A., and Goldstein, T. Adversarial attacks on copyright detection systems. In *International Conference on Machine Learning*, pp. 8307–8315. PMLR, 2020.

Schuhmann, C., Beaumont, R., Vencu, R., Gordon, C., Wightman, R., Cherti, M., Coombes, T., Katta, A., Mullis, C., Wortsman, M., et al. Laion-5b: An open large-scale dataset for training next generation image-text models. *arXiv preprint arXiv:2210.08402*, 2022.

Sitawarin, C., Tramèr, F., and Carlini, N. Preprocessors matter! realistic decision-based attacks on machine learning systems. *arXiv preprint arXiv:2210.03297*, 2022.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. Intriguing properties of neural networks. In *International Conference on Learning Representations*, 2014.

Tramèr, F., Dupré, P., Rusak, G., Pellegrino, G., and Boneh, D. Adversarial: Perceptual ad blocking meets adversarial machine learning. In *ACM SIGSAC Conference on Computer and Communications Security*, 2019.

Usmanova, I., Krause, A., and Kamgarpour, M. Safe non-smooth black-box optimization with application to policy search. In *Learning for Dynamics and Control*, pp. 980–989. PMLR, 2020.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Le Scao, T., Gugger, S., Drame, M., Lhoest, Q., and Rush, A. M. Transformers: State-of-the-Art Natural Language Processing. pp. 38–45. Association for Computational Linguistics, 10 2020. URL https://www.aclweb.org/anthology/2020.emnlp-demos.6.

Yuan, K., Tang, D., Liao, X., Wang, X., Feng, X., Chen, Y., Sun, M., Lu, H., and Zhang, K. Stealthy porn: Understanding real-world adversarial images for illicit online promotion. In *2019 IEEE Symposium on Security and Privacy (SP)*, pp. 952–966. IEEE, 2019.

A. Related Work

Threat models for ML evasion attacks. Modeling realistic ML evasion attacks is challenging (Gilmer et al., 2018; Apruzzese et al., 2022). 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 (Biggio et al., 2013), copyright systems (Saadatpanah et al., 2020), or online content blockers (Tramèr et al., 2019; Yuan et al., 2019). These works either assume white-box model access, or use black-box *transfer* attacks. The latter are perfectly stealthy (they make no bad queries) but have limited success rates.

Detecting decision-based attacks. Chen et al. (2020b) and Li et al. (2022) detect decision-based attacks by monitoring sequences of user queries. We aim to evade a more fundamental form of monitoring that any security-critical system likely uses: flagging and banning users who issue many "bad" queries.

Stealthy score-based attacks. *Score-based attacks*, which query a model's confidence scores (Chen et al., 2017), also issue many bad queries. Designing stealthy score-based attacks is similar to the problem of "safe black-box optimization" in reinforcement learning (Usmanova et al., 2020).

B. Details on Experimental Setup

B.1. Datasets and Models

ImageNet. We run the attacks against a ResNet-50 (He et al., 2016) classifier trained on ImageNet (Deng et al., 2009). We use the model weights provided as part of the torchvision library (Paszke et al., 2019), 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. We mark a query as bad if it is classified into the class of the original input.

ImageNet-Dogs. To capture more realistic security-critical scenarios, we consider a variety of binary classification tasks that aim to separate "good" from "bad" data. As a toy benchmark, we create a binary classification task from ImageNet by considering as "bad" the images belonging to classes of dog breeds (i.e., the classes with indices included in the range $[151, 268]$) and as "good" 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 (Kingma & Ba, 2014) 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 "bad" images, we only attack the images in the validation set that are correctly classified as "bad" (i.e., as dogs) by the fine-tuned model.

ImageNet-NSFW. As mentioned in Section 5.2, we also evaluate the attacks on the NSFW content detector shared by Schuhmann et al. (2022). This classifier takes as input CLIP (Radford et al., 2021) embeddings of images and outputs a confidence in $[0, 1]$. We use the CLIP implementation provided by the HuggingFace Transformers library (Wolf et al., 2020) 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 (Prabhu & Birhane, 2020)). When attacking the model, we consider an attack to be successful if the confidence of the detector drops below 0.5. 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 NSFW content. We consider a query bad if scored with 4 or 5.

B.2. Attack Hyper-parameters and implementation details

BOUNDARY ATTACK. We use the official implementation[3], which is part of Foolbox (Rauber et al., 2017), with default hyper-parameters on all tasks.

HOPSKIPJUMP. We use the official implementation.[4] Following Sitawarin et al. (2022), we set $\texttt{gamma} = 10{,}000$ (this hyper-parameter is used to determine the binary search threshold), as this gives better results.

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

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

RAYS and STEALTHY RAYS. We use the official implementation.[5] 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 5.4, 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 5 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.[6] Following Sitawarin et al. (2022), we set $\beta = 10^{-2}$ (this hyper-parameter is used to determine the binary search threshold).

For STEALTHY OPT, 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 $\texttt{dist}$, we start the search at the point at distance $(1+\gamma) \cdot \texttt{dist}$ along θ', for $\gamma > 0$. If this point is not misclassified (i.e., the query is bad), 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 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 bad 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.[7] Following Sitawarin et al. (2022), 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 is the version with a full line-search, and $k = 2.5$.

B.3. 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 line search. We wrap all the attack implementations in a common set-up for which we use PyTorch (Paszke et al., 2019). The code can be found at the following URL: `https://github.com/ethz-spylab/realistic-adv-examples`. 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://github.com/ethz-spylab/realistic-adv-examples/releases/tag/v0.1`.

C. Details on Decision-based Attacks

In this section, we provide some additional detail on how existing decision-based attacks work, and how they spend their bad queries.

As explained in Section 4.1, existing decision-based attacks optimize over some *adversarial direction* $\theta \in [0,1]^d$ by repeatedly: (1) computing the boundary distance $\texttt{dist}$ from x along θ; (2) computing an update direction δ; and (3) picking a step-size α, in order to perform an update step $\theta \leftarrow \theta + \alpha \cdot \delta$.

We can thus split each attack iteration into three phases:

- `projBoundary`: given the original input (x, y) and a search direction θ, 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 ℓ_p distance between x and x_b, i.e.,

[5] `https://github.com/uclaml/RayS/blob/29bc17/RayS.py`
[6] `https://github.com/cmhcbb/attackbox/blob/65a82f/attack/OPT_attack.py`
[7] `https://github.com/cmhcbb/attackbox/blob/65a82f/attack/Sign_OPT.py`

Table 1. Median number of queries for each attack to reach a median ℓ_2 distance of 10 and median ℓ_∞ distance of $^8/_{255}$ on untargeted ImageNet. We report the total number of attack queries Q_{total}, and of "bad" queries Q_{bad} (queries that get classified as the class that the attacker wants to evade).

Norm	Attack	Total Queries Q_{total}	Bad Queries Q_{bad}
ℓ_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%)
ℓ_∞	HOPSKIPJUMP	3,591	1,789 (50%)
	RAYS	328	244 (74%)

Table 2. Where do decision-based attacks spend their queries? We run untargeted attacks against a ResNet-50 on ImageNet (see Section 5.1 for details). For each attack, we report the fraction of queries used in `checkAdv` or `getDist` routines, and the fraction of bad queries in each routine.

		`checkAdv`		`getDist`	
Norm	Attack	all	frac. bad	all	frac. bad
ℓ_2	BOUNDARY	100%	84%	0%	–
	OPT	2%	50%	98%	52%
	SIGN-OPT	77%	52%	23%	57%
	HOPSKIPJUMP	93%	55%	7%	43%
ℓ_∞	HOPSKIPJUMP	92%	50%	8%	50%
	RAYS	36%	67%	64%	78%

`dist` $\leftarrow \|x - x_b\|_p$.

- `updateDir`: This phase searches for an update direction δ to be applied to the search direction θ.
- `stepSize`: This phase selects a step-size α for an update to the search direction θ.

We now describe how different attacks instantiate these generic phases and how they use the `checkAdv` and `getDist` routines in each phase.

BOUNDARY ATTACK. The original decision-based attack of Brendel et al. (2017) 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 θ (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 δ 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 ℓ_∞ 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 θ.

`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 θ.

`updateDir`: The attack estimates the gradient of the distance to the boundary along the search direction θ. 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

Table 3. 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	$\texttt{checkAdv} \cdot n$	–	`checkAdv`	$\texttt{checkAdv} \cdot (n+1)$
OPT	`getDist`	$\texttt{getDist} \cdot n$	$\texttt{getDist} \cdot m$	$\texttt{getDist} \cdot (n+m+1)$
SIGN-OPT	`getDist`	$\texttt{checkAdv} \cdot n$	$\texttt{getDist} \cdot m$	$\texttt{checkAdv} \cdot n$ + $\texttt{getDist} \cdot (m+1)$
HOPSKIPJUMP	`getDist`	$\texttt{checkAdv} \cdot n$	`getDist`	$\texttt{checkAdv} \cdot n$ + $\texttt{getDist} \cdot 2$
RAYS	`getDist`	–	`checkAdv`	`checkAdv` + `getDist`

each. The estimated gradient is then:

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

The attack uses n calls to `getDist` to compute the boundary distance along each random direction.

`stepSize`: OPT computes the step-size α 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 θ. But the two views, and their implementations, are equivalent. Both attacks use a 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

$$\delta \leftarrow \frac{1}{n} \sum_{i=1}^{n} z_i \cdot r_i \; , \tag{2}$$

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 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 δ 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 δ, albeit with a geometric backtracking search instead of a binary search.

Summary of attacks. In Table 2, we show how many bad 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 "bad"). In Table 3 summarizes how different attacks implement the three generic attack phases `projBoundary`, `updateDir` and `stepSize` in each attack iteration. We distinguish here between the two routines called by the attacks, `checkAdv` and `getDist` defined in Appendix C.

D. Convergence Rates of Stealthy Attacks

Prior work has analyzed the convergence rate of SGD with the zero-order gradient estimation schemes used in SIGN-OPT and OPT (Liu et al., 2018; Cheng et al., 2019). We can use these results to prove that the gradient estimation of our

STEALTHY OPT attack is asymptotically more efficient (in terms of bad queries) than the non-stealthy gradient estimation used by `sign` and HOPSKIPJUMP.

Let $g(\theta)$ be the distance to the boundary along the direction θ, 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. (2018) (Theorem 2))**.** *Assume g has gradients that are L-Lipshitz 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 $\mathcal{O}(T^2/d)$ bad queries.*

Theorem 2 (Adapted from Cheng et al. (2019) (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 $\mathcal{O}(T^2 d)$ bad queries.*

The convergence rate of OPT is thus at least as good as that of SIGN-OPT,[8] but OPT's gradient estimator with line searches requires a factor d^2 fewer bad queries. The same asymptotic result as for SIGN-OPT holds for the similar estimator used by HOPSKIPJUMP.

Proof of Theorem 1. Liu et al. (2018) 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., 2018)). 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 bad query if we use a line search. This yields the number of bad queries in the theorem (T iterations with $\frac{T}{d}$ bad queries per iteration). □

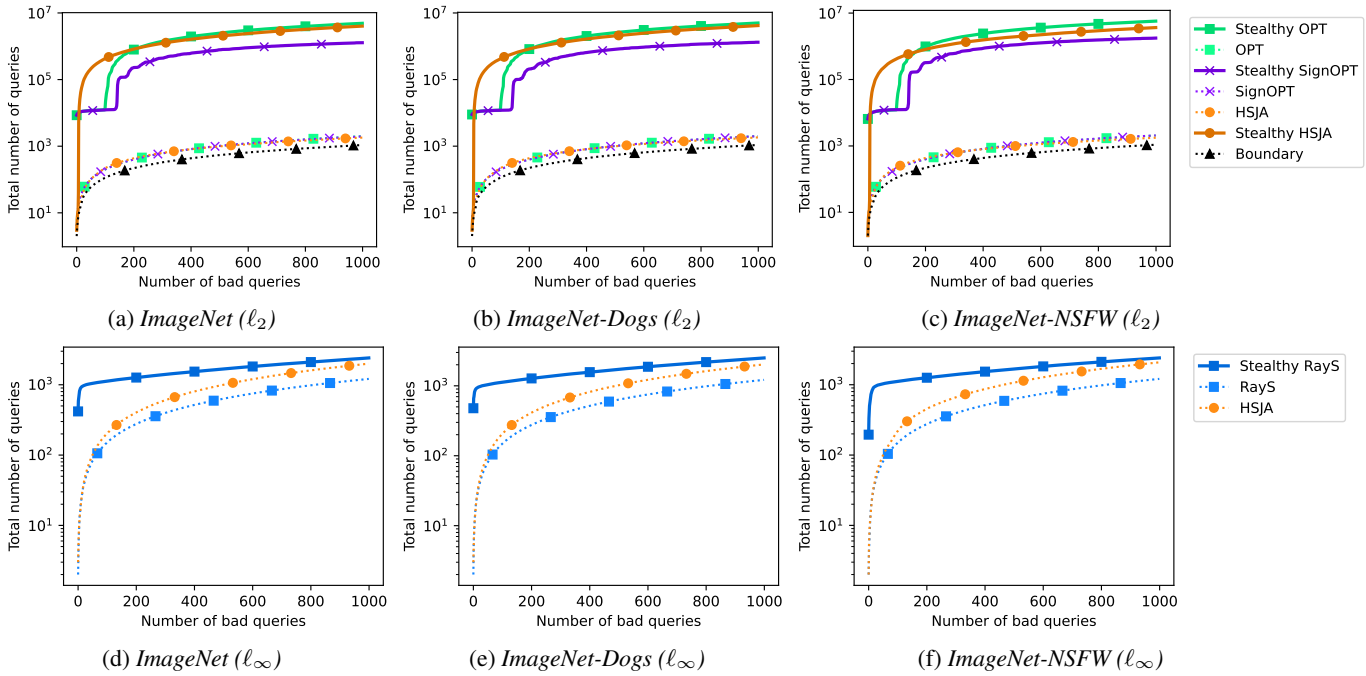
Proof of Theorem 2. Cheng et al. (2019) show that SIGN-OPT's gradient estimator yields a convergence rate of $\mathbb{E}[\|\nabla g(x)\|_2]=\mathcal{O}(\sqrt{d/T})+\mathcal{O}\left(d/\sqrt{Q'}\right)$ (see Theorem 3.1 in (Cheng et al., 2019)). To balance the two convergence terms, we set $Q'=Td$. To perform Q' evaluations of $\texttt{sign}(g(\theta+r_i)-g(\theta))$, one call to `getDist` and Q' calls to `checkDir` are required. Each `checkDir` call makes a single query to the model f, i.e., $1/2$ bad queries on average. This yields the number of bad queries in the theorem (T iterations with $\frac{Td}{2}$ bad queries per iteration). □

[8] Note that Cheng et al. (2019) provide a bound on the gradient norm, while Liu et al. (2018) 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 $\mathbb{E}[\|\nabla g(x)\|_2^2]\geq(\mathbb{E}[\|\nabla g(x)\|_2])^2=\mathcal{O}(d/T)$.

E. Additional Figures and Tables

Table 4. Ablation on stealthy attack components. For attacks on ImageNet, we show the relative number of bad queries and good queries (lower is better) compared to the original non-stealthy attack, to achieve a median ℓ_2 perturbation of 10, or ℓ_∞ perturbation of $^8/_{255}$.

Attack	Ablation	Bad queries reduction (higher is better)	Good queries increase (lower is better)
HopSkipJump	with OPT gradient estimation	1.56×	1418.72×
	with OPT gradient 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×



(a) *ImageNet* (ℓ_2) (b) *ImageNet-Dogs* (ℓ_2) (c) *ImageNet-NSFW* (ℓ_2)

(d) *ImageNet* (ℓ_∞) (e) *ImageNet-Dogs* (ℓ_∞) (f) *ImageNet-NSFW* (ℓ_∞)

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

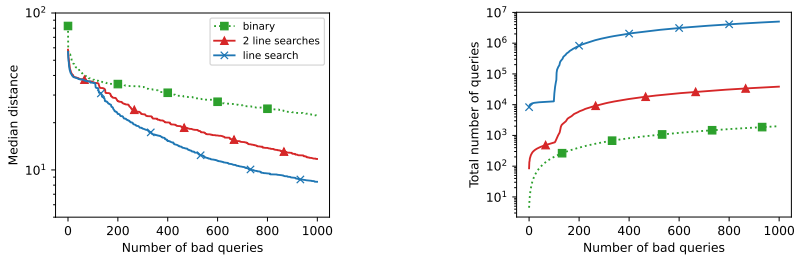


Figure 7. Trade-offs between good and bad queries for different search strategies in the STEALTHY OPT attack. A full line-search makes one bad query and up to 10,000 good queries. The version with two searches makes two bad queries and up to $2 \cdot 100$ good queries.

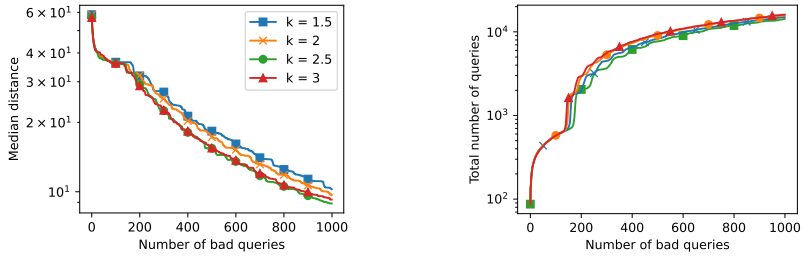


Figure 8. Influence of the hyper-parameter k in the STEALTHY SIGN-OPT attack (the reduction in number of gradient estimation queries per iteration). The best results are obtained with $k = 2.5$.

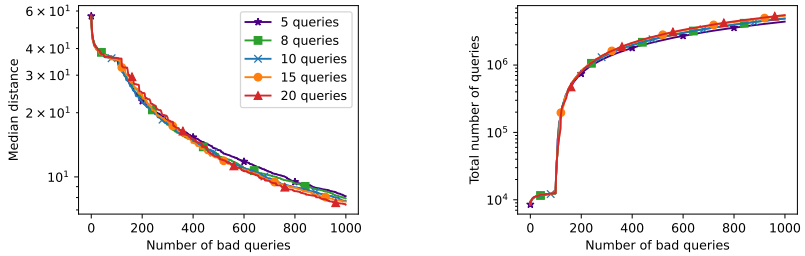


Figure 9. Influence of the number of directions computed for the gradient estimation in the STEALTHY OPT attack. The best results are obtained with $q = 10$, which is the original value from Cheng et al. (2018).

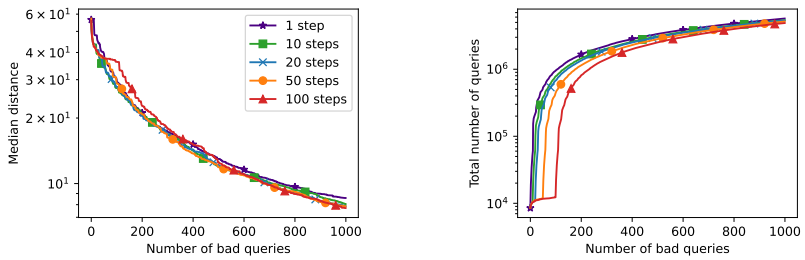
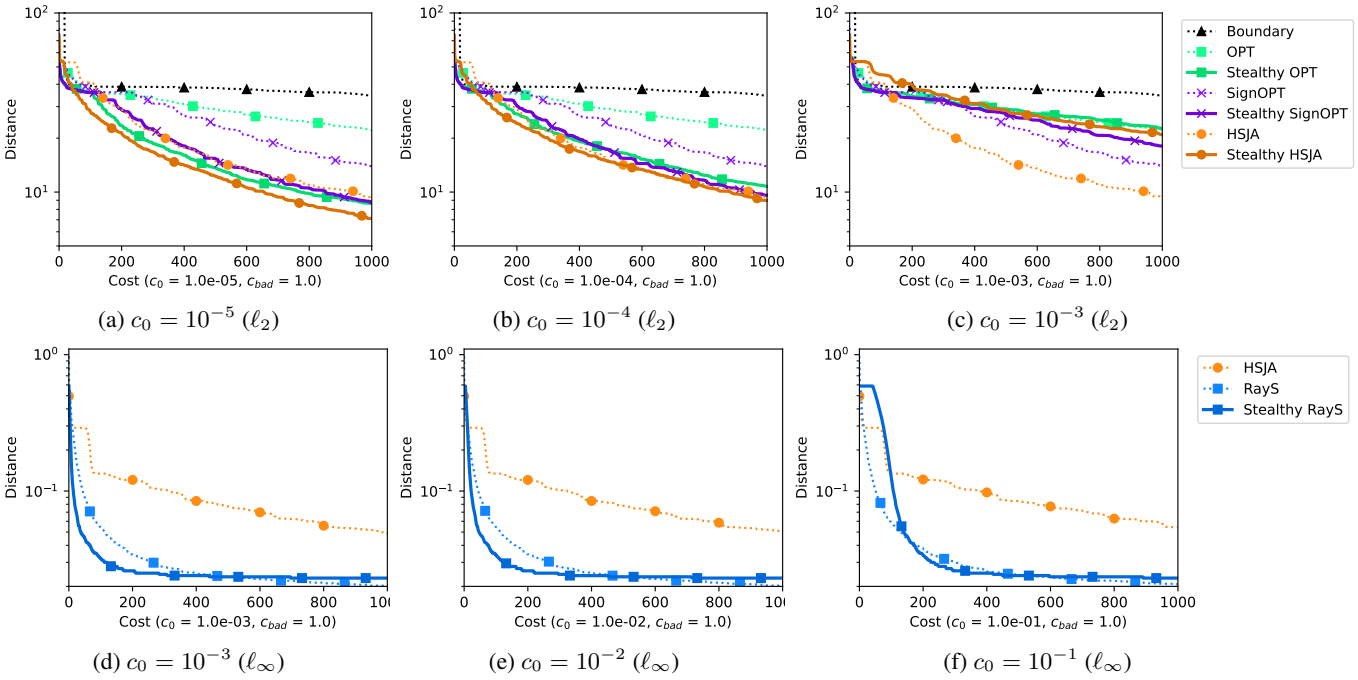


Figure 10. Influence of the number of directions tested for the initialization in the STEALTHY OPT attack. The best results are obtained with $n = 100$, when considering a larger number of queries, even though the difference between the different values is small.



(a) $c_0 = 10^{-5}$ (ℓ_2) (b) $c_0 = 10^{-4}$ (ℓ_2) (c) $c_0 = 10^{-3}$ (ℓ_2)

(d) $c_0 = 10^{-3}$ (ℓ_∞) (e) $c_0 = 10^{-2}$ (ℓ_∞) (f) $c_0 = 10^{-1}$ (ℓ_∞)

Figure 11. Costs trade-offs of various decision-based attacks on ImageNet, for different asymmetric costs of good and bad queries. We show how the attack cost varies for different values of the base query cost c_0, at a fixed cost $c_{\text{bad}} = 1$ for bad queries. The advantage given by the stealthy attacks is reduced when the relative cost of good queries increases.