SIGN BITS ARE ALL YOU NEED FOR BLACK-BOX ATTACKS

Anonymous authors
Paper under double-blind review

ABSTRACT

We present a novel black-box adversarial attack algorithm with state-of-the-art model evasion rates for query efficiency under \( \ell_\infty \) and \( \ell_2 \) metrics. It exploits a sign-based, rather than magnitude-based, gradient estimation approach that shifts the gradient estimation from continuous to binary black-box optimization. It adaptively constructs queries to estimate the gradient, one query relying upon the previous, rather than re-estimating the gradient each step with random query construction. Its reliance on sign bits yields a smaller memory footprint and it requires neither hyperparameter tuning or dimensionality reduction. Further, its theoretical performance is guaranteed and it can characterize adversarial subspaces better than white-box gradient-aligned subspaces. On two public black-box attack challenges and a model robustly trained against transfer attacks, the algorithm’s evasion rates surpass all submitted attacks. For a suite of published models, the algorithm is 3.8× less failure-prone while spending 2.5× fewer queries versus the best combination of state of art algorithms. For example, it evades a standard MNIST model using just 12 queries on average. Similar performance is observed on a standard IMAGENET model with an average of 579 queries.

1 INTRODUCTION

Problem. Deep Neural Networks (DNNs) are vulnerable to adversarial examples, which are malicious inputs designed to fool the model’s prediction—see (Biggio and Roli, 2018) for a comprehensive, recent overview of adversarial examples. Research on generating these malicious inputs started in the white-box setting, where access to the gradients of the models is assumed. Since the gradient points to the direction of steepest ascent, an input can be perturbed along the gradient’s direction to maximize the network’s loss, thereby potentially causing misclassification under class prediction, e.g. with images, or evasion under detection, e.g. with malware. The assumption of access to the underlying gradient does not however reflect real world scenarios. Attack algorithms under a more realistic, restrictive black-box threat model, which assumes access to predictions in lieu of gradients, are therefore studied. Central to their approaches is estimating the gradient. To estimate the magnitudes and signs of the gradient, the community at large has formulated a continuous optimization problem of \( O(n) \) complexity where \( n \) is the input dimensionality. Most recently work has sought to reduce this complexity by means of data-/time-dependent priors (Ilyas et al., 2019). In this paper, we take a different tact and reduce the central problem to just estimating the signs of the gradients. Our intuition arises from observing that estimating the sign of the top 30% gradient coordinates by magnitude is enough to achieve a rough misclassification rate of 70%. Figure 1 reproducing Ilyas et al. (2019) illustrates this observation for the MNIST dataset—see Appendix A for other datasets. Therefore our goal is to recover the sign of the gradient with high query efficiency so we can use it to generate adversarial examples as effective as those generated by full gradient estimation approaches.

Related Work. We organize the related work in two themes, namely Adversarial Example Generation and Sign-Based Optimization. The literature of the first theme primarily divides into white-box and black-box settings. The white-box setting, while not the focus of this work, follows from the works of Biggio et al. (2013) and Goodfellow et al. (2015) who introduced the Fast Gradient Sign Method (FGSM), including several methods to produce adversarial examples for various learning tasks and threat perturbation constraints (Carlini and Wagner, 2017; Moosavi-Dezfooli et al., 2016; Hayes and Danezis, 2017; Al-Dujaili et al., 2018; Kurakin et al., 2017; Shamir et al., 2019). Turning to the black-box setting and iterative optimization schemes, Narodytska and Kasiviswanathan (2017), without
using any gradient information, use a naive policy of perturbing random segments of an image to generate adversarial examples. Bhagoji et al. (2017) reduce the dimensions of the feature space using Principal Component Analysis (PCA) and random feature grouping, before estimating gradients. Chen et al. (2017) introduce a principled approach by using gradient based optimization. They employ finite differences, a zeroth-order optimization means, to estimate the gradient and then use it to design a gradient-based attack. While this approach successfully generates adversarial examples, it is expensive in how many times the model is queried. Ilyas et al. (2018) substitute traditional finite differences methods with Natural Evolutionary Strategies (NES) to obtain an estimate of the gradient. Tu et al. (2018) provide an adaptive random gradient estimation algorithm that balances query counts and distortion, and introduces a trained auto-encoder to achieve attack acceleration. Ilyas et al. (2019) extend this line of work by proposing the idea of gradient priors and bandits: Bandits_{TD}. Our work contrasts with the general approach of these works in two ways: a) We focus on estimating the sign of the gradient and investigate whether this estimation suffices to efficiently generate adversarial examples. b) The above methods employ random sampling in constructing queries to the model while our construction is adaptive. Another approach involves learning adversarial examples for one model (with access to its gradient information) to transfer them against another (Liu et al. (2016), Papernot et al. (2017)). Alternately, Xiao et al. (2018) use a Generative Adversarial Network (GAN) to generate adversarial examples which are based on small norm-bounded perturbations. These methods involve learning on a different model, which is expensive, and not amenable to comparison with setups—including ours—that directly query the model of interest.

**Sign-Based Optimization.** In the context of general-purpose continuous optimization methods, sign-based stochastic gradient descent was studied in both zeroth- and first-order setups. In the latter, Bernstein et al. (2018) analyzed signSGD, a sign-based Stochastic Gradient Descent, and showed that it enjoys a faster empirical convergence than SGD in addition to the cost reduction of communicating gradients across multiple workers. Liu et al. (2019) extended signSGD to zeroth-order setup with the ZO-SignSGD algorithm. ZO-SignSGD (Liu et al. 2019) was shown to outperform NES against a black-box model on MNIST. These approaches use the sign of the gradient (or its zero-order estimate) to achieve better convergence, whereas our approach both estimates and uses the sign of the gradient.

**Contributions.** We present the following contributions at the intersection of adversarial machine learning and black-box (zeroth-order) optimization: 1) We exploit the separability property of the directional derivative of the loss function of the model under attack in the direction of \{±1\}^n vectors, to propose a divide-and-conquer, adaptive, memory-efficient algorithm, we name SignHunter, to estimate the gradient sign bits. 2) We provide a worst-case theoretical guarantee on the number of queries required by SignHunter to perform at least as well as FGSM (Goodfellow et al. 2015), which has access to the model’s gradient. To our knowledge, no black-box attack from the literature offers a similar performance guarantee. 3) We evaluate our approach on a rigorous set of experiments on both, standard and adversarially hardened models. All other previous works on this topic have published their results on a subset of the datasets and threat models we experimentally validate in this work. Through these experiments, we demonstrate that SignHunter’s adaptive search for the gradient sign allows it to craft adversarial examples within a mere fraction of the theoretical number of queries thus outperforming FGSM and state-of-the-art black-box attacks. 4) We release a software framework to systematically benchmark adversarial black-box attacks, including SignHunter’s, on MNIST, CIFAR10, and IMAGENET models in terms of success rate, query count, and other metrics. 5) We demonstrate how SignHunter can be used to characterize adversarial cones in a black-box setup and in doing so, highlight the gradient masking effect.

**Notation.** Let \( n \) denote the dimension of data point \( \mathbf{x} \). Denote a hidden \( n \)-dimensional binary code by \( q^* \). That is, \( q^* \in \mathcal{H} \equiv \{-1, +1\}^n \). Further, denote the directional derivative of some function \( f \) at a
point \( x \) in the direction of a vector \( v \) by \( D_v f(x) \equiv v^T \nabla_x f(x) \) which often can be approximated by the finite difference method. That is, for \( \delta > 0 \), we have

\[
D_v f(x) = v^T \nabla_x f(x) \approx \frac{f(x + \delta v) - f(x)}{\delta}.
\]  

(1)

Let \( \Pi_S(\cdot) \) be the projection operator onto the set \( S \), \( B_p(x, \epsilon) \) be the \( \ell_p \) ball of radius \( \epsilon \) around \( x \).

2 \hspace{1cm} \textbf{Gradient Estimation}

At the heart of black-box adversarial attacks is generating a perturbation vector to slightly modify the original input \( x \) so as to fool the network prediction of its true label \( y \). Put differently, an adversarial example \( x' \) maximizes the network’s loss \( L(x', y) \) but still remains \( \epsilon \)-close to the original input \( x \). Although the loss function \( L \) can be non-concave, gradient-based techniques are often very successful in crafting an adversarial example [Madry et al. (2017)]. That is, setting the perturbation vector as a step in the direction of \( \nabla_x L(x, y) \). Consequently, the bulk of black-box attack methods try to estimate the gradient by querying an oracle that returns, for a given input/label pair \((x, y)\), the value of the network’s loss \( L(x, y) \), consulting prediction or classification accuracy. Using only such value queries, the basic approach relies on the finite difference method to approximate the directional derivative (Eq. 1) of the function \( L \) at the input/label pair \((x, y)\) in the direction of a vector \( v \), which corresponds to \( v^T \nabla_x L(x, y) \). With \( n \) linearly independent vectors \( \{v_i^T \nabla_x L(x, y) = d_i\}_{1 \leq i \leq n} \), one can construct a linear system of equations to recover the full gradient. Clearly, this approach’s query complexity is \( O(n) \), which can be prohibitively expensive for large \( n \) (e.g., \( n = 268,203 \) for the IMAGENET dataset). Recent works try to mitigate this issue by exploiting data- and/or time-dependent priors [Tu et al. (2018), Ilyas et al. (2018, 2019)]. However, the queries are not adaptive, they are constructed based on i.i.d. random vectors \( \{v_i\} \). They fail to make use of the past queries’ responses to construct the new query and recover the full gradient more efficiently. As stated in the introduction, we solve the smaller problem of gradient sign estimation with adaptive queries based on the observation that simply leveraging (noisy) sign bits of the gradient yields successful attacks—see Figure 1.

\textbf{Definition 1. (Gradient Sign Estimation Problem)} For an input/label pair \((x, y)\) and a loss function \( L \), let \( g^* = \nabla_x L(x, y) \) be the gradient of \( L \) at \((x, y)\) and \( q^* = \text{sign}(g^*) \in H \) be the sign bit vector of \( g^* \). Then the goal of the gradient sign estimation problem is to find a binary vector \( q \in H \) maximizing the directional derivative of \( L \) at \((x, y)\) from a limited number of (possibly adaptive) function value queries \( L(x', y) \).

3 \hspace{1cm} \textbf{A Method for Estimating Sign of the Gradient from Adaptive Queries}

Our goal is to estimate the gradient sign bits of the loss function \( L \) of the model under attack at an input/label pair \((x, y)\) from a limited number of loss value adaptive queries \( L(x', y) \). To this end, we examine the basic concept of directional derivatives that has been employed in recent black-box adversarial attacks. Based on the definition of the directional derivative (Eq. 1), the following can be stated.

\textbf{Property 1 (Separability of} \( D_q L(x, y) \)). The directional derivative \( D_q L(x, y) \) of the loss function \( L \) at an input/label pair \((x, y)\) in the direction of a binary code \( q \) is separable. That is, \( \max_{q \in H} D_q L(x, y) = \max_{q \in H} q^T g^* = \sum_{i=1}^{n} \max_{q_i \in \{-1, +1\}} q_i g_i^* \).

(3)

\footnote{Without loss of generality, we encode the sign bit vector in \( H \equiv \{-1, +1\}^n \) rather than \( \{0, 1\}^n \). This is a common representation in sign-related literature. Note that the standard sign function has the range of \( \{-1, 0, +1\} \). Here, we use the non-standard definition (Zhai (2018) whose range is \( \{-1, +1\} \). This follows from the observation that DNNs’ gradients with respect to their inputs are not sparse (Ilyas et al. (2019) Appendix B.1).}

\footnote{The maximization follows from \( D_q L(x, y) = q^T g^* \), which is maximized when \( q = q^* = \text{sign}(g^*) \).}
This reformulates the gradient sign estimation problem from single $n$-dimensional to $n$ 1-dimensional binary black-box optimization problems, reducing the search space of sign bits from $2^n$ to $2n$. Subsequently, one could recover the gradient sign bits with $n + 2$ queries as follows: i. Start with an arbitrary sign vector $q$ and compute the directional derivative $D_q L(x, y)$. Using Eq. 1, this requires two queries: $L(x + \delta q, y)$ and $L(x, y + \delta q)$. ii. For the remaining $n$ queries, flip $q$’s bits (coordinates) one by one and compute the corresponding directional derivative $D_q L(x, y)$ and revert those otherwise. This, however, still suffers from the $O(n)$ complexity of full gradient estimation methods. Further, each query recovers at most one sign bit and the natural question to ask is: can we recover more sign bits per query?

Consider the case where all the gradient coordinates have the same magnitude, i.e., $\{|g_i^q|\}_{1 \leq i \leq n} = 1$, and let the initial guess $q_1$ have $r$ correct bits and $n - r$ wrong ones. Instead of flipping its bits sequentially, we can flip them all at once to get $q_2 = -q_1$. If $D_{q_1} L(x, y) \geq D_{q_2} L(x, y)$, then we retain $q_2$ as our best guess with $n - r$ correct bits, otherwise $q_1$ remains. In either cases, with three queries, we will recover $\max(r, n - r)$ sign bits. One can think of this flip/revert procedure as one of majority voting by the guess’s coordinates on whether they agree with their gradient sign’s counterparts. To see this, let $|g_i^q| = 1$ for all $i$, then the condition $D_{q_1} L(x, y) \geq D_{q_2} L(x, y)$ can be written as $n - r - r \geq r - n + r \implies n \geq 2r$. If the agree votes $r$ are less than half of the total votes $n$, then $q_2$ is retained. Besides flipping all the coordinates, one can employ the same procedure iteratively on a subset (chunk) of the coordinates $[q_{j}, \ldots, q_{j+n}]$ of the guess vector $q$, recovering $\max(r, n_i - r_i)$ sign bits, where $n_i$ and $r_i$ is the length of the $i$th chunk and the number of its correct signs, respectively.

While the magnitudes of gradient coordinates may not have the same value as assumed in the previous example; through empirical evaluation (see Appendix F), we found them to be concentrated. Consequently and with high probability, their votes on retaining or reverting chunks of sign flips are weighted (by their corresponding gradient magnitude) similarly. That said, if we are at a chunk where the distribution of the gradient coordinate magnitudes is uniform, then the flip/revert procedure could favor recovering few sign coordinates with large magnitude counterparts over many sign coordinates with small magnitude counterparts. From our experiments on the noisy FGSM, this still suffices to generate adversarial examples: an attack with 30% correct sign bits (that correspond to the top gradient coordinates magnitudes) is more effective than an attack with 50% correct arbitrary sign bits as shown in Figure 1. Put differently, we would like to recover as many sign bits as possible with as few queries as possible. However, if we can only recover few, they should be those that correspond to coordinates with large gradient magnitude. This notion is in line with the flip/revert procedure.

We employ the above observation in a divide-and-conquer search which we refer to as SignHunter. As outlined in Algorithm 1, the technique starts with an initial guess of the sign vector $q_1$ ($s$ in Algorithm 1). It then proceeds to flip the sign of all the coordinates to get a new sign vector $q_2$, and revert the flips if the loss oracle returned a value $L(x + \delta q_2, y)$ (or equivalently the directional derivative) less than the best obtained so far $L(x + \delta q_1, y)$. SignHunter applies the same rule to the first half of the coordinates, the second half, the first quadrant, the second quadrant, and so on. For a search space of dimension $n$, SignHunter needs $2^{[\log(n)+1]} - 1$ sign flips to complete its search. If the query budget is not exhausted by then, you can update $x$ with the recovered signs and restart the procedure at the updated point with a new starting code $s$. If we start with a sign vector whose Hamming distance to the optimal sign vector $q^*$ is $n/2$: agreeing with $q^*$ in the first half of coordinates. In this case, SignHunter needs just four queries to recover the entire sign vector.

### Algorithm 1

**SignHunter**

$g : \mathcal{H} \rightarrow \mathbb{R}$: the black-box function to be maximized over the binary hypercube $\mathcal{H} \equiv \{-1, +1\}^n$

```python
def init(g):
    i ← 0, h ← 0
    g ← g
    s ← $U(\mathcal{H})$ // e.g., $[+1, \ldots, +1]$
    done ← false
    g_best ← $\infty$

def is_done() :
    return done

def step():
    c_len ← $[n/2^h]$  
    s[i*c_len:(i+1)*c_len] *= -1
    if g(s) ≥ g_best:
        g_best ← g(s)
    else:
    s[i*c_len:(i+1)*c_len] *= -1
    increment i
    if i == $2^h$:
        i ← 0, increment h
        if h == $[\log_2(n)] + 1$:
            done ← true

def get_current_sign_estimate() :
    return s
```

---

Under review as a conference paper at ICLR 2020
independent of \( n \), whereas the sequential bit flipping still require \( n + 2 \) queries. In the next theorem, we show that SignHunter is guaranteed to perform at least as well as FGSM with \( \mathcal{O}(n) \) oracle queries. Up to our knowledge, no such guarantees exist for any black-box attack from the literature.

**Theorem 1.** (Optimality of SignHunter) Given \( 2^{\lfloor \log(n)+1 \rfloor} \) queries and that the directional derivative is well approximated by the finite-difference (Eq. 7), SignHunter is at least as effective as FGSM (Goodfellow et al., 2015) in crafting adversarial examples.

The proof can be found in Appendix B. Theorem 1 provides an upper bound on the number of queries required for SignHunter to recover the gradient sign bits, and perform as well as FGSM. In practice (as will be shown in our experiments), SignHunter crafts adversarial examples with a small fraction of this upper bound. The rationale here is that we do not need to recover the sign bits exactly; we rather need a fast convergence to an adversarially helpful sign vector \( s \). In our setup, we use the best sign estimation obtained \( s \) so far in a similar fashion to FGSM, whereas full-gradient estimation approaches often employ an iterative scheme of \( T \) steps within the perturbation ball \( B_\epsilon(p) \), calling the gradient estimation routine in every step leading to a search complexity of \( nT \). Instead, our gradient sign estimation routine runs at the top level of our adversarial example generation procedure. Further, SignHunter is amenable to parallel hardware architecture and has a smaller memory footprint (just sign bits) and thus can carry out attacks in batches more efficiently. Crafting black-box adversarial attacks with SignHunter is outlined in Algorithm 2.

### 4 Experiments

We evaluate SignHunter and compare it with established algorithms from the literature: ZO-SignSGD [Liu et al., 2019], NES [Ilyas et al., 2018], and BanditsRd [Ilyas et al., 2019] in terms of effectiveness in crafting (without loss of generality) untargeted black-box adversarial examples. To highlight SignHunter’s adaptive query construction, we introduce a variant of Algorithm 2 named Rand. At every iteration, Rand’s sign vector is sampled uniformly from \( \mathcal{H} \). Both \( \ell_\infty \) and \( \ell_2 \) threat models are considered on the MNIST, CIFAR10, and IMAGENET datasets.

**Experiments Setup.** Our experiment setup is similar to [Ilyas et al., 2019]. Each attacker is given a budget of 10,000 oracle queries per attack attempt and is evaluated on 1000 images from the test sets of MNIST, CIFAR10, and the validation set of IMAGENET. We did not find a standard practice for setting the perturbation bound \( \epsilon \). For the \( \ell_\infty \) threat model, we use [Madry et al., 2017]’s bound for MNIST and [Ilyas et al., 2019]’s bounds for both CIFAR10 and IMAGENET. For the \( \ell_2 \) threat model, [Ilyas et al., 2019]’s bound is used for IMAGENET. MNIST’s bound is set based on the sufficient distortions observed in [Liu et al., 2019], which are smaller than the one used in [Madry et al., 2017].

We use the observed bound in [Cohen et al., 2019] for CIFAR10. We show results based on standard models—i.e., models that are not adversarially hardened. For MNIST and CIFAR10, the naturally trained models from [Madry et al., 2017]’s MNIST and CIFAR10 challenges are used. For IMAGENET, TensorFlow’s Inception (v3) model is used. The loss oracle returns the cross-entropy loss. We evaluate SignHunter with

\[
\text{Algorithm 2 Black-Box Adversarial Example Generation with SignHunter}
\]

- **\( x_{init} \): input to be perturbed, \( y_{init} \): \( x_{init} \)’s true label, \( B_\epsilon(p) \): \( \ell_p \) perturbation ball of radius \( \epsilon \)
- **\( L \): loss function of the model under attack**

1: \( \delta \leftarrow \epsilon \) // set finite-difference probe to perturbation bound
2: \( x_o \leftarrow x_{init} \)
3: Define the function \( g \) as
4: \( g(q) = L(\Pi_{B_\epsilon(p)}(x_{o} + \delta q), y_{init}) - L(x_{o}, y_{init}) \)
5: SignHunter.init()  
6: // \( C() \) returns top class
7: while \( C(x) = y_{init} \) do  
8: \( s \leftarrow \text{SignHunter.get_current_sign_estimate()} \)
9: \( x \leftarrow \Pi_{B_\epsilon(p)}(x_{o} + \delta s) \)
10: if SignHunter.is_done() then
11: \( x_o \leftarrow x \)
12: Define the function \( g \) as in Line 3 (with \( x_o \) update)
13: SignHunter.init()
14: return \( x \)

**Hyperparameters Setup.** While SignHunter does not have any hyperparameters, to fairly compare it with the other algorithms, we tuned their hyperparameters starting with the default values reported by the corresponding authors. The finite difference probe \( \delta \) for SignHunter is set to the perturbation bound \( \epsilon \).

\[ \text{Algorithm 2 Black-Box Adversarial Example Generation with SignHunter} \]

- **\( x_{init} \): input to be perturbed, \( y_{init} \): \( x_{init} \)’s true label, \( B_\epsilon(p) \): \( \ell_p \) perturbation ball of radius \( \epsilon \)
- **\( L \): loss function of the model under attack**

1: \( \delta \leftarrow \epsilon \) // set finite-difference probe to perturbation bound
2: \( x_o \leftarrow x_{init} \)
3: Define the function \( g \) as
4: \( g(q) = L(\Pi_{B_\epsilon(p)}(x_{o} + \delta q), y_{init}) - L(x_{o}, y_{init}) \)
5: SignHunter.init()  
6: // \( C() \) returns top class
7: while \( C(x) = y_{init} \) do  
8: \( s \leftarrow \text{SignHunter.get_current_sign_estimate()} \)
9: \( x \leftarrow \Pi_{B_\epsilon(p)}(x_{o} + \delta s) \)
10: if SignHunter.is_done() then
11: \( x_o \leftarrow x \)
12: Define the function \( g \) as in Line 3 (with \( x_o \) update)
13: SignHunter.init()
14: return \( x \)

\[ \text{That is, replace Line 8 in Algorithm 2 by } s \sim U(\mathcal{H}). \]
Figure 2: Performance of black-box attacks in the $\ell_\infty$ and $\ell_2$ perturbation constraint. The plots show the average number of queries used per successful image for each attack when reaching a specified success rate.

bound $\epsilon$ as it is used for both computing the finite difference and crafting the adversarial examples—see Line 1 in Algorithm 2. This parameter-free aspect of SignHunter offers a robustness advantage over algorithms which require expert hyperparameter tuning. Details on the hyperparameter setup are available in Appendix C.

Results. Figure 2 shows the trade-off between the success (evasion) rate and the mean number of queries (of the successful attacks, per convention) needed to generate an adversarial example for the MNIST, CIFAR10, and IMAGENET classifiers under the $\ell_\infty$ and $\ell_2$ perturbation constraints. These plots indicate the average number of queries required for a desired success rate. We observe the following: For any given success rate, SignHunter dominates the previous state of the art approaches in all settings except the IMAGENET $\ell_2$ setup, where BanditsTD shows a better query efficiency when the desired success rate is roughly greater than 0.35. This is all the more remarkable because BanditsTD exploits tiles, a data-dependent prior, searching over $50 \times 50 \times 3$ dimensions for IMAGENET, while SignHunter searches over the explicit data $299 \times 299 \times 3$ dimensions: $36 \times 36 \times 3$ more dimensions.

$\ell_\infty$ vs. $\ell_2$ Perturbation Threat. In view of BanditsTD’s advantage, SignHunter is remarkably efficient in the $\ell_\infty$ setup, achieving a 100% success rate using—on average—just 12 queries per image against the MNIST classifier! In the $\ell_2$ setup, SignHunter’s performance degrades—yet it still outperforms the other algorithms. This is expected, since SignHunter perturbs all the coordinates with the same magnitude and the $\ell_2$ perturbation bound $\epsilon_2$ for all the datasets in our experiments is set such that $\epsilon_2/\sqrt{n}$ is significantly less than the $\ell_\infty$ perturbation bound $\epsilon_\infty$. Take the case of MNIST ($n = 28 \times 28$), where $\epsilon_\infty = 0.3$ and $\epsilon_2 = 3$. For SignHunter, the $\ell_2$ setup is equivalent to an $\ell_\infty$ perturbation bound of $3/28 \approx 0.1$. The employed $\ell_2$ perturbation bounds give the state of the art—continuous optimization based—approaches more perturbation options. For instance, it is possible for NES to perturb just one pixel in an MNIST image by a magnitude of 3; two pixels by a magnitude of $3/\sqrt{2} \approx 2.1$ each; ten pixels by a magnitude of $3/\sqrt{10} \approx 0.9$ each, etc. On the other hand, the binary optimization view of SignHunter limits it to always perturb all $28 \times 28$ pixels by a magnitude of $3/28 \approx 0.1$. Despite its fewer degrees of freedom, SignHunter maintains its effectiveness in the $\ell_2$ setup. The plots can also be interpreted as a sensitivity assessment of SignHunter as $\epsilon$ gets smaller going from $\ell_\infty$ to the $\ell_2$ perturbation threat.

SignHunter vs FGSM. The performance of SignHunter is in line with Theorem 1 when compared with the performance of FGSM (the noisy FGSM at $k = 100\%$ in Figures 1 and 2 of Appendix A) in both $\ell_\infty$ and $\ell_2$ setups across all datasets. For instance, FGSM has a failure rate of 0.32 for CIFAR10 $\ell_2$ (Appendix A, Figure 2 (b)), while SignHunter achieves a failure rate of 0.21 with

---

Tabulated summaries of these plots can be found in Appendix D—Tables 7–9. We also plot the classifier’s loss and the gradient estimation quality (in terms of Hamming distance and cosine similarity) averaged over all the images as a function of the number of queries in Figures 4–6 of Appendix D.
692.39 < 2n = 2 \times 3 \times 32 \times 32 = 6144 \text{ queries (Appendix D, Table 8)}. Note that for IMAGENET, SignHunter outperforms FGSM with a query budget of 10,000 queries, a fraction of the theoretical number of queries required \( 2n = 536,406 \) to perform at least as well. Incorporating SignHunter in an iterative framework of perturbing the data point \( x \) till the query budget is exhausted (Lines 10 to 14 in Algorithm\(^2\)) supports the observation in white-box settings that iterative FGSM—or Projected Gradient Descent (PGD)—is stronger than FGSM \( ^{[\text{Madry et al.}, 2017; Al-Dujaili et al., 2018]} \). This is evident by the upticks in SignHunter’s performance on the MNIST \( \ell_2 \) case (Appendix D, Figure 4), which happens after every iteration (after every other \( 2 \times 28 \times 28 \) queries).

**Gradient Estimation.** Plots of the Hamming similarity capture the number of recovered sign bits, while plots of the average cosine similarity capture the value of Eq.\(^2\). Both SignHunter and Bandits\(_{TD}\) consistently optimize both metrics. In general, SignHunter (Bandits\(_{TD}\)) converges faster especially on the Hamming(cosine) metric as it is estimating the signs (signs and magnitudes) compared to Bandits\(_{TD}\)’s full gradient (SignHunter’s gradient sign) estimation. This is most obvious in the IMAGENET \( \ell_2 \) setup (Appendix D, Figure 6). Note that once an attack is successful, the estimated gradient sign at that point is used for the rest of the plot. This explains why, in the \( \ell_\infty \) settings, SignHunter’s plot does not improve compared to its \( \ell_2 \) counterpart, as most of the attacks are successful in the very first few queries made to the loss oracle and no further refined estimation is required. Another possible reason is that the gradient direction can be very local and does not capture the global loss landscape compared to SignHunter’s estimation. More on this is discussed in Section\(^6\).

**SignHunter vs. Rand.** Given these results, one could argue that SignHunter is effective, because it maximally perturbs datapoints to the vertices of their perturbation balls. However, Rand’s poor performance does not support this argument and highlights the effectiveness of SignHunter’s adaptive query construction. Except for MNIST and CIFAR10 \( \ell_\infty \) settings, Rand performs worse than the full-gradient estimation approaches, although it perturbs datapoints similar to SignHunter. Overall, SignHunter is 3.8× less failure-prone than the state-of-the-art approaches combined, and spends over all the images (successful and unsuccessful attacks) 2.5× less queries.\(^6\)

5 **SignHunter vs. Defenses**

To complement Section\(^4\), we evaluate SignHunter against adversarial training, a way to improve the robustness of DNNs \( ^{[\text{Madry et al.}, 2017]} \). Specifically, we attacked the secret models used in public challenges for MNIST and CIFAR10. For IMAGENET, we used ensemble adversarial training, a method that argues security against black-box attacks based on transferability \( ^{[\text{Tramer et al.}, 2017a]} \). Appendix E reports the same metrics used in Section\(^4\) as well as a tabulated summary for the results discussed below.

**Public MNIST Black-Box Attack Challenge.** In line with the challenge setup, 10,000 test images were used with an \( \ell_\infty \) perturbation bound of \( \epsilon = 0.3 \). Although the secret model is released, we treated it as a black box similar to our experiments in Section\(^4\). No maximum query budget was specified, so we set it to 5,000 queries. This is equal to the number of iterations given to a PGD attack in the white-box setup of the challenge: 100-steps with 50 random restarts. SignHunter’s attacks resulted in the lowest model accuracy of 91.47%, outperforming all the submitted attacks to the challenge, with an average number of queries of 233 per successful attack. Note that the attacks submitted to the challenge are based on transferability and do not query the model of interest. On the other hand, the most powerful white-box attack by \( ^{[\text{Wang et al.}, 2018]} \)—as of May 15, 2019—resulted in a model accuracy of 88.42%. Further, a PGD attack with 5,000 back-propagations achieves 89.62% in contrast to SignHunter’s 91.47% with just 5,000 forward-propagations.

**Public CIFAR10 Black-Box Attack Challenge.** This challenge setup is similar to the above, but with an \( \ell_\infty \) perturbation bound of \( \epsilon = 8 \). SignHunter’s attacks resulted in the lowest model accuracy of 47.16%, outperforming all the submitted attacks to the challenge, with an average number of queries of 569 per successful attack. Similar to the MNIST challenge, all the submitted attacks are based on transferability. On the other hand, the most powerful white-box attack by \( ^{[\text{Zheng et al.}, 2018]} \)—as of May 15, 2019—resulted in a model accuracy of 44.71%. Further, a PGD attack

---

\(^3\)We define perturbation vertices as extreme points of the ball \( B_p(x, \epsilon) \). That is, \( x \pm \epsilon \), where \( \epsilon = \epsilon_p \) when \( p = \infty \) and \( \epsilon_\infty = \epsilon / \sqrt{n} \) when \( p = 2 \).

\(^4\)The number of queries spent is computed based on Tables 7–9 of Appendix D as \( (1 - \text{fail}_\text{rate}) \times \text{avg}_\text{#_queries} + \text{fail}_\text{rate} \times 10,000 \).
Assuming a black-box threat model, we studied the problem of generating adversarial examples with its gradient. SignHunter (2017a) still exhibits the gradient masking (note here, this is important in adversarial settings, where the loss landscape is analyzed in the vicinity of the vectors maximally aligned with the gradient sign). v3-GAAS finds more orthogonal adversarial vectors than GAAS, thereby providing a better characterization of the space of adversarial examples in the vicinity of a point, albeit without a white-box access to the models.

Ensemble Adversarial Training on IMAGENET. In line with Tramèr et al. (2017a), we set $\epsilon = 0.0625$ and report the v3adv-ens4 model’s misclassification over 10,000 random images from IMAGENET’s validation set. After 20 queries, SignHunter achieves a top-1 error of 40.61% greater than the 33.4% rate of a series of black-box attacks (including PGD with 20 iterations) transferred from a substitute model. With 1000 queries, SignHunter breaks the model’s robustness with a top-1 error of 90.75%!

6 Characterizing Adversarial Cones with SignHunter

Estimating the size of adversarial cones, the space of adversarial examples in the vicinity of a point, for a model has been a topic of interest by the machine learning community Tramèr et al. (2017a), Ma et al. (2018), Lu et al. (2018). The Gradient-Aligned Adversarial Subspace (GAAS) method Tramèr et al. (2017b) provides an approximation of the adversarial cone dimensionality by finding a set of orthogonal perturbations of norm $\epsilon$ that are all adversarial with respect to the model. By linearizing the model’s loss function, this is reduced to finding orthogonal vectors that are maximally aligned with its gradient $g^*$—or its gradient sign $q^*$ in the $\ell_\infty$ setup Tramèr et al. (2017a). In Figure 3, we reproduce (Tramèr et al. 2017a, Fig. 2) and show that aligning the orthogonal vectors with SignHunter’s estimation (we refer to this approach as SAAS) instead of aligning them with the gradient (GAAS) results in a better approximation of the adversarial cone for the two IMAGENET models considered earlier, even when the number of queries given to SignHunter is just a fraction of the dimensionality $n$. Through its query-efficient finite-difference sign estimation, SignHunter is able to quickly capture the larger-scale variation of the loss landscape in the point’s neighborhood, rather than the infinitesimal point-wise variation that the gradient provides, which can be very local. This is important in adversarial settings, where the loss landscape is analyzed in the vicinity of the point Moosavi-Dezfooli et al. (2018), Tramèr et al. (2017a). One interesting observation at $k = 1$ (note here, $r_1 = q^*$) across all $\epsilon$ is that GAAS finds adversarial directions for fewer points against the v3adv-ens4 Model than the naturally trained model v3, whereas SAAS reports similar probability of adversarial directions for both. This contrast suggests that ensemble adversarial training Tramèr et al. (2017a) still exhibits the gradient masking effect, where the gradient poorly approximates the global loss landscape.

7 Conclusion

Assuming a black-box threat model, we studied the problem of generating adversarial examples for neural nets and proposed the gradient sign estimation problem as the core challenge in crafting these examples. We formulate the problem as a binary black-box optimization one: maximizing
the directional derivative in the direction of \(\{\pm 1\}^n\) vectors, approximated by the finite difference of the queries’ loss values. The separability property of the directional derivative helped us devise **SignHunter**, a query-efficient, hyperparameter-free divide-and-conquer algorithm with a small memory footprint that is guaranteed to perform at least as well as FGSM after \(O(n)\) queries. No similar guarantee is found in the literature. In practice, **SignHunter** needs a mere fraction of this number of queries to craft adversarial examples. The algorithm is one of its kind to construct adaptive queries instead of queries that are based on i.i.d. random vectors. Robust to gradient masking, **SignHunter** can also be used to estimate the dimensionality of adversarial cones. Moreover, **SignHunter** achieves the highest evasion rate on two public black-box attack challenges and breaks a model that argues robustness against substitute-model attacks.

**REFERENCES**


http://proceedings.mlr.press/v80/bernstein18a.html


APPENDIX A. NOISY FGSM

This section shows the performance of the noisy FGSM on standard models (described in Section 1 of the main paper) on the MNIST, CIFAR10 and IMAGENET datasets. In Figure 4, we consider the \( \ell_\infty \) threat perturbation constraint. Figure 5 reports the performance for the 2 setup. Similar to Ilyas et al. (2019), for each \( k \) in the experiment, the top \( k \) percent of the signs of the coordinates—chosen either randomly (random-\( k \)) or by the corresponding magnitude \( |\partial L(x,y)/\partial x| \) (top-\( k \))—are set correctly, and the rest are set to \(-1\) or \(+1\) at random. The misclassification rate shown considers only images that were correctly classified (with no adversarial perturbation). In accordance with the models’ accuracy, there were 987, 962, and 792 such images for MNIST, CIFAR10, and IMAGENET out of the sampled 1000 images, respectively. These figures also serve as a validation for Theorem 3 of the main paper when compared to SignHunter’s performance shown in Appendix D.

Figure 4: Misclassification rate of three neural nets (for (a) MNIST, (b) CIFAR10, and (c) IMAGENET, respectively) on the noisy FGSM’s adversarial examples as a function of correctly estimated coordinates of \( \text{sign}(\nabla_x f(x,y)) \) on random 1000 images from the corresponding evaluation dataset, with the maximum allowed \( \ell_\infty \) perturbation \( \epsilon \) being set to 0.3, 12, and 0.05, respectively. Across all the models, estimating the sign of the top 30\% gradient coordinates (in terms of their magnitudes) is enough to achieve a misclassification rate of \( \sim 70\% \). Note that Plot (c) is similar to Ilyas et al. (2019)’s Figure 1, but it is produced with TensorFlow rather than PyTorch.

Figure 5: Misclassification rate of three neural nets (for (a) MNIST, (b) CIFAR10, and (c) IMAGENET, respectively) on the noisy FGSM’s adversarial examples as a function of correctly estimated coordinates of \( \text{sign}(\nabla_x f(x,y)) \) on random 1000 images from the corresponding evaluation dataset, with the maximum allowed \( \ell_2 \) perturbation \( \epsilon \) being set to 3, 127, and 5, respectively. Compared to Figure 4, the performance on MNIST and CIFAR10 drops significantly.
APPENDIX B. PROOFS FOR THEOREMS IN THE MAIN PAPER

Theorem 1. (Optimality of SignHunter) Given $2^\lceil\log(n)+1\rceil$ queries and that the directional derivative is well approximated by the finite-difference (Eq. 1 in the main paper), SignHunter is at least as effective as FGSM [Goodfellow et al., 2015] in crafting adversarial examples.

Proof. Based on the separability property of the directional derivative, the $i$th coordinate of the gradient sign vector can be recovered as follows: construct two binary codes $u$ and $v$ such that only their $i$th bit is different. Therefore, we have

$$q_i^* = \text{sign}(g_i^*) = \begin{cases} u_i & \text{if } D_u L(x, y) > D_v L(x, y), \\ v_i & \text{otherwise.} \end{cases} \quad (4)$$

From the definition of SignHunter, this is carried out for all the $n$ coordinates after $2^\lceil\log(n)+1\rceil$ queries. Put it differently, after $2^\lceil\log(n)+1\rceil$ queries, SignHunter has flipped every coordinate alone recovering its sign exactly as shown in Eq. (4) above. Therefore, the gradient sign vector is fully recovered, and one can employ the FGSM attack to craft an adversarial example. Note that this is under the assumption that our finite difference approximation of the directional derivative (Eq. 1 in the main paper) is good enough (or at least rank-preserving). \qed
APPENDIX C. EXPERIMENTS SETUP

This section outlines the experiments setup. To ensure a fair comparison among the considered algorithms, we did our best in tuning their hyperparameters. Initially, the hyperparameters were set to the values reported by the corresponding authors, for which we observed suboptimal performance. We made use of a synthetic concave loss function to efficiently tune the algorithms for each dataset $\times$ perturbation constraint combination. The performance curves on the synthetic loss function using the tuned values of the hyperparameters did show consistency with the reported results from the literature. For instance, we noted that ZO-SignSGD converges faster than NES, and that Bandits$_{TD}$ outperformed the rest of the algorithms towards the end of query budget. Further, in our adversarial examples generation experiments, we observed failure rate and query efficiency in line with the algorithms’ corresponding papers—e.g., compare the performance of Bandits$_{TD}$ and NES in Table 9 of Appendix D with (Ilyas et al., 2019) Table 1). That said, we invite the community to provide their best tuned attacks.

Note that SignHunter does not have any hyperparameters to tune. The finite difference probe $\delta$ for SignHunter is set to the perturbation bound $\epsilon$ as it is used for for both computing the finite difference and crafting the adversarial examples—see Line 1 in Algorithm 2 of the main paper. This parameter-free setup of SignHunter offers a robust edge over the state-of-the-art black-box attacks, which often require expert knowledge to carefully tune their parameters.

Table 2 describes the general setup for the attacks. Table 1 lists the sources of the models we attacked in this work, while Tables 3, 4, 5, and 6 outline the algorithms’ hyperparameters. Figure 6 shows the performance of the considered algorithms on a synthetic concave loss function after tuning their hyperparameters. All experiments were run on a CUDA-enabled NVIDIA Tesla V100 16GB.

A possible explanation of SignHunter’s superb performance is that the synthetic loss function is well-behaved in terms of its gradient given an image. That is, most of gradient coordinates share the same sign, since pixels tend to have the same values and the optimal value for all the pixels is the same $x_{\text{min}} + x_{\text{max}} / 2$. Thus, SignHunter will recover the true gradient sign with as few queries as possible (recall the example in Section 3 of the main paper). Moreover, given the structure of the synthetic loss function, the optimal loss value is always at the boundary of the perturbation region; the boundary is where SignHunter samples its perturbations.

### Table 1: Source of attacked models.

<table>
<thead>
<tr>
<th>Model</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>MNIST models</td>
<td><a href="https://github.com/MadryLab/mnist_challenge">https://github.com/MadryLab/mnist_challenge</a></td>
</tr>
<tr>
<td>CIFAR10 models</td>
<td><a href="https://github.com/MadryLab/cifar10_challenge">https://github.com/MadryLab/cifar10_challenge</a></td>
</tr>
</tbody>
</table>

### Table 2: General setup for all the attacks

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>MNIST</th>
<th>CIFAR10</th>
<th>IMAGENET</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\epsilon$ (allowed perturbation)</td>
<td>0.3 3 12 127 0.05 5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Max allowed queries</td>
<td>10000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Evaluation/Test set size</td>
<td>10000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data (pixel value) Range</td>
<td>[0,1] [0,255] [0,1]</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Table 3: Hyperparameters setup for NES

<table>
<thead>
<tr>
<th>Hyperparameter</th>
<th>Value</th>
<th>MNIST</th>
<th>CIFAR10</th>
<th>IMAGENET</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\delta$ (finite difference probe)</td>
<td>0.1</td>
<td>0.1</td>
<td>2.55</td>
<td>2.55</td>
</tr>
<tr>
<td>$\eta$ (image $\ell_p$ learning rate)</td>
<td>0.1</td>
<td>1</td>
<td>2</td>
<td>127</td>
</tr>
<tr>
<td>$q$ (number of finite difference estimations per step)</td>
<td>10</td>
<td>20</td>
<td>4</td>
<td>100</td>
</tr>
</tbody>
</table>
The plots above show the average performance of the tuned attacks on the synthetic loss function $L(x, y) = - (x - x^*)^2 (x - x^*)$, where $x^* = \frac{\min_{x \in \mathcal{X}}}{2 \times \max_{x \in \mathcal{X}}}$ using a query limit of 1000 queries for each image. Note that in all, BanditsTD outperforms both NES and ZO-SignSGD. Also, we observe the same behavior reported by [Liu et al. (2019)] on the fast convergence of ZO-SignSGD compared to NES. We did not tune SignHunter; it does not have any tunable parameters.
**APPENDIX D. RESULTS OF ADVERSARIAL BLACK-BOX EXAMPLES GENERATION**

This section shows results of our experiments in crafting adversarial black-box examples on standard models in the form of tables and performance traces, namely Figures 7, 8, and 9 and Tables 7, 8, and 9.

Table 7: Summary of attacks effectiveness on MNIST under $\ell_\infty$ and $\ell_2$ perturbation constraints, and with a query limit of 10,000 queries. The Failure Rate $\in [0, 1]$ column lists the fraction of failed attacks over 1000 images. The Avg. # Queries column reports the average number of queries made to the loss oracle only over successful attacks.

<table>
<thead>
<tr>
<th>Attack</th>
<th>$\ell_\infty$</th>
<th>$\ell_2$</th>
<th>$\ell_\infty$</th>
<th>$\ell_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>BanditsTD</td>
<td>0.68</td>
<td>0.59</td>
<td>328.00</td>
<td>673.16</td>
</tr>
<tr>
<td>NES</td>
<td>0.63</td>
<td>0.63</td>
<td>235.07</td>
<td>361.42</td>
</tr>
<tr>
<td>Rand</td>
<td>0.33</td>
<td>0.96</td>
<td>847.77</td>
<td>1144.74</td>
</tr>
<tr>
<td>SignHunter</td>
<td>0.00</td>
<td>0.04</td>
<td>11.06</td>
<td>1064.22</td>
</tr>
<tr>
<td>ZOSignSGD</td>
<td>0.63</td>
<td>0.75</td>
<td>157.00</td>
<td>881.08</td>
</tr>
</tbody>
</table>

Table 8: Summary of attacks effectiveness on CIFAR10 under $\ell_\infty$ and $\ell_2$ perturbation constraints, and with a query limit of 10,000 queries. The Failure Rate $\in [0, 1]$ column lists the fraction of failed attacks over 1000 images. The Avg. # Queries column reports the average number of queries made to the loss oracle only over successful attacks.

<table>
<thead>
<tr>
<th>Attack</th>
<th>$\ell_\infty$</th>
<th>$\ell_2$</th>
<th>$\ell_\infty$</th>
<th>$\ell_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>BanditsTD</td>
<td>0.95</td>
<td>0.39</td>
<td>432.24</td>
<td>1201.85</td>
</tr>
<tr>
<td>NES</td>
<td>0.37</td>
<td>0.67</td>
<td>312.57</td>
<td>496.99</td>
</tr>
<tr>
<td>Rand</td>
<td>0.20</td>
<td>0.89</td>
<td>422.16</td>
<td>1018.17</td>
</tr>
<tr>
<td>SignHunter</td>
<td>0.07</td>
<td>0.21</td>
<td>121.00</td>
<td>692.39</td>
</tr>
<tr>
<td>ZOSignSGD</td>
<td>0.37</td>
<td>0.80</td>
<td>161.28</td>
<td>528.35</td>
</tr>
</tbody>
</table>

Table 9: Summary of attacks effectiveness on IMAGENET under $\ell_\infty$ and $\ell_2$ perturbation constraints, and with a query limit of 10,000 queries. The Failure Rate $\in [0, 1]$ column lists the fraction of failed attacks over 1000 images. The Avg. # Queries column reports the average number of queries made to the loss oracle only over successful attacks.

<table>
<thead>
<tr>
<th>Attack</th>
<th>$\ell_\infty$</th>
<th>$\ell_2$</th>
<th>$\ell_\infty$</th>
<th>$\ell_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>BanditsTD</td>
<td>0.07</td>
<td>0.11</td>
<td>1010.05</td>
<td>1635.55</td>
</tr>
<tr>
<td>NES</td>
<td>0.26</td>
<td>0.42</td>
<td>1536.19</td>
<td>1993.86</td>
</tr>
<tr>
<td>Rand</td>
<td>0.72</td>
<td>0.93</td>
<td>688.77</td>
<td>418.02</td>
</tr>
<tr>
<td>SignHunter</td>
<td>0.02</td>
<td>0.23</td>
<td>578.56</td>
<td>1985.55</td>
</tr>
<tr>
<td>ZOSignSGD</td>
<td>0.23</td>
<td>0.52</td>
<td>1654.98</td>
<td>931.15</td>
</tr>
</tbody>
</table>
Figure 7: Performance curves of attacks on MNIST for \( \ell_\infty \) (first column) and \( \ell_2 \) (second column) perturbation constraints. Plots of Avg. Loss row reports the loss as a function of the number of queries averaged over all images. The Avg. Hamming Similarity row shows the Hamming similarity of the sign of the attack’s estimated gradient \( \hat{g} \) with true gradient’s sign \( q^* \), computed as \( 1 - ||\text{sign}(\hat{g}) - q^*||_H/n \) and averaged over all images. Likewise, plots of the Avg. Cosine Similarity row show the normalized dot product of \( \hat{g} \) and \( g^* \) averaged over all images. The Success Rate row reports the attacks’ cumulative distribution functions for the number of queries required to carry out a successful attack up to the query limit of 10,000 queries. The Avg. # Queries row reports the average number of queries used per successful image for each attack when reaching a specified success rate: the more effective the attack, the closer its curve is to the bottom right of the plot.
Figure 8: Performance curves of attacks on CIFAR10 for $\ell_\infty$ (first column) and $\ell_2$ (second column) perturbation constraints. Plots of Avg. Loss row reports the loss as a function of the number of queries averaged over all images. The Avg. Hamming Similarity row shows the Hamming similarity of the sign of the attack’s estimated gradient $\hat{g}$ with true gradient’s sign $q^*$, computed as $1 - ||\text{sign}(\hat{g}) - q^*||_H/n$ and averaged over all images. Likewise, plots of the Avg. Cosine Similarity row show the normalized dot product of $\hat{g}$ and $g^*$ averaged over all images. The Success Rate row reports the attacks’ cumulative distribution functions for the number of queries required to carry out a successful attack up to the query limit of 10,000 queries. The Avg. # Queries row reports the average number of queries used per successful image for each attack when reaching a specified success rate: the more effective the attack, the closer its curve is to the bottom right of the plot.
Figure 9: Performance curves of attacks on IMAGENET for $\ell_\infty$ (first column) and $\ell_2$ (second column) perturbation constraints. Plots of Avg. Loss row reports the loss as a function of the number of queries averaged over all images. The Avg. Hamming Similarity row shows the Hamming similarity of the sign of the attack’s estimated gradient $\hat{g}$ with true gradient’s sign $q^*$, computed as $1 - ||\text{sign}(\hat{g}) - q^*||_H/n$ and averaged over all images. Likewise, plots of the Avg. Cosine Similarity row show the normalized dot product of $\hat{g}$ and $g^*$ averaged over all images. The Success Rate row reports the attacks’ cumulative distribution functions for the number of queries required to carry out a successful attack up to the query limit of 10,000 queries. The Avg. # Queries row reports the average number of queries used per successful image for each attack when reaching a specified success rate: the more effective the attack, the closer its curve is to the bottom right of the plot.
APPENDIX E. PUBLIC BLACK-BOX CHALLENGE RESULTS

This section shows results of our experiments in crafting adversarial black-box examples on adversarially trained models in the form of tables and performance traces, namely Tables 10, 11, and Figure 10.

Table 10: Leaderboard of the MNIST black-box challenge. Adapted from the challenge’s website— as of May 15, 2019.

<table>
<thead>
<tr>
<th>Black-Box Attack</th>
<th>Model Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>SignHunter (Algorithm 2 in the main paper)</td>
<td>91.47%</td>
</tr>
<tr>
<td>[Xiao et al. (2018)]</td>
<td>92.76%</td>
</tr>
<tr>
<td>PGD against 3 independently &amp; adversarially trained copies of the network</td>
<td>93.54%</td>
</tr>
<tr>
<td>FGSM on the CW loss for model B from [Tramèr et al., 2017a]</td>
<td>94.36%</td>
</tr>
<tr>
<td>FGSM on the CW loss for the naturally trained public network</td>
<td>96.08%</td>
</tr>
<tr>
<td>PGD on the cross-entropy loss for the naturally trained public network</td>
<td>96.81%</td>
</tr>
<tr>
<td>Attack using Gaussian Filter for selected pixels on the adversarially trained</td>
<td>97.33%</td>
</tr>
<tr>
<td>public network</td>
<td></td>
</tr>
<tr>
<td>FGSM on the cross-entropy loss for the adversarially trained public network</td>
<td>97.66%</td>
</tr>
<tr>
<td>PGD on the cross-entropy loss for the adversarially trained public network</td>
<td>97.79%</td>
</tr>
</tbody>
</table>

Table 11: Leaderboard of the CIFAR10 black-box challenge. Adapted from the challenge’s website— as of May 15, 2019.

<table>
<thead>
<tr>
<th>Black-Box Attack</th>
<th>Model Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>SignHunter (Algorithm 2 in the main paper)</td>
<td>47.16%</td>
</tr>
<tr>
<td>PGD on the cross-entropy loss for the adversarially trained public network</td>
<td>63.39%</td>
</tr>
<tr>
<td>PGD on the CW loss for the adversarially trained public network</td>
<td>64.38%</td>
</tr>
<tr>
<td>FGSM on the CW loss for the adversarially trained public network</td>
<td>67.25%</td>
</tr>
<tr>
<td>FGSM on the CW loss for the naturally trained public network</td>
<td>85.23%</td>
</tr>
</tbody>
</table>

Table 12: Top 1 Error Percentage. The numbers between brackets are computed on 10,000 images from the validation set. The rest are from [Tramèr et al., 2017a] Table 4.

<table>
<thead>
<tr>
<th>Model</th>
<th>clean</th>
<th>Max. Black-box</th>
<th>SignHunter after 20 queries</th>
<th>SignHunter after 1000 queries</th>
</tr>
</thead>
<tbody>
<tr>
<td>v3 adv-ens4</td>
<td>24.2 (26.73)</td>
<td>33.4</td>
<td>(40.61)</td>
<td>(90.75)</td>
</tr>
<tr>
<td>Madry’s Lab (MNIST)</td>
<td>Madry’s Lab (CIFAR10)</td>
<td>Ensemble Adv. Training (IMAGENET)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------------------</td>
<td>-----------------------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td><img src="image1.png" alt="Graph" /></td>
<td><img src="image2.png" alt="Graph" /></td>
<td><img src="image3.png" alt="Graph" /></td>
<td></td>
<td></td>
</tr>
<tr>
<td><img src="image4.png" alt="Graph" /></td>
<td><img src="image5.png" alt="Graph" /></td>
<td><img src="image6.png" alt="Graph" /></td>
<td></td>
<td></td>
</tr>
<tr>
<td><img src="image7.png" alt="Graph" /></td>
<td><img src="image8.png" alt="Graph" /></td>
<td><img src="image9.png" alt="Graph" /></td>
<td></td>
<td></td>
</tr>
<tr>
<td><img src="image10.png" alt="Graph" /></td>
<td><img src="image11.png" alt="Graph" /></td>
<td><img src="image12.png" alt="Graph" /></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 10: Performance curves of attacks on the public black-box challenges for MNIST (first column), CIFAR10 (second column) and IMAGENET (third column). Plots of Avg. Loss row reports the loss as a function of the number of queries averaged over all images. The Avg. Hamming Similarity row shows the Hamming similarity of the sign of the attack’s estimated gradient $\hat{g}$ with true gradient’s sign $q^*$, computed as $1 - ||\text{sign}(\hat{g}) - q^*||_H/n$ and averaged over all images. Likewise, plots of the Avg. Cosine Similarity row show the normalized dot product of $\hat{g}$ and $q^*$ averaged over all images. The Success Rate row reports the attacks’ cumulative distribution functions for the number of queries required to carry out a successful attack up to the query limit of 5,000 queries for MNIST and CIFAR10 (1,000 queries for IMAGENET). The Avg. # Queries row reports the average number of queries used per successful image for each attack when reaching a specified success rate: the more effective the attack, the closer its curve is to the bottom right of the plot.
Appendix F. Histogram of Gradient Coordinates’ Magnitudes

This section illustrates our experiment on the distribution of the magnitudes of gradient coordinates as summarized in Figure 11. How to read the plots: Consider the first histogram in Plot (a) from below; it corresponds to the 1000th image from the sampled MNIST evaluation set, plotting the histogram of the values \(\{|\partial L(x, y)/\partial x_i|\}_{1 \leq i \leq n}\), where the MNIST dataset has dimensionality \(n = 784\). These values are in the range \([0, 0.002]\). Overall, the values are fairly concentrated—with exceptions, in Plot (e) for instance, the magnitudes of the \(\sim 400^{th}\) image’s gradient coordinates are spread from 0 to \(\sim 0.055\).

<table>
<thead>
<tr>
<th>MNIST</th>
<th>CIFAR10</th>
<th>IMAGENET</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="a" alt="gradient coordinates" /></td>
<td><img src="b" alt="gradient coordinates" /></td>
<td><img src="c" alt="gradient coordinates" /></td>
</tr>
<tr>
<td><img src="d" alt="gradient coordinates" /></td>
<td><img src="e" alt="gradient coordinates" /></td>
<td><img src="f" alt="gradient coordinates" /></td>
</tr>
</tbody>
</table>

Figure 11: Magnitudes of gradient coordinates are concentrated: Plots (a), (b), and (c) show histograms of the magnitudes of gradient coordinates of the loss function \(L(x, y)\) with respect to the input point (image) \(x\) for MNIST, CIFAR10, and IMAGENET neural net models over 1000 images from the corresponding evaluation set, respectively. Plots (d), (e), (f) show the same but at input points (images) sampled randomly within \(B_\infty(x, \epsilon)\): the \(\ell_\infty\)-ball of radius \(\epsilon = 0.3, 12,\) and \(0.05\) around the images in Plots (a), (b), and (c), respectively.