There are No Bit Parts for Sign Bits in Black-Box Attacks

Anonymous Authors[1]

Abstract

We present a black-box adversarial attack algorithm which sets new state-of-the-art model evasion rates for query efficiency in the ℓ_∞ and ℓ_2 metrics, where only loss-oracle access to the model is available. On two public black-box attack challenges, the algorithm achieves the highest evasion rate, surpassing all of the submitted attacks. Similar performance is observed on a model that is secure against substitute-model attacks. For standard models trained on the MNIST, CIFAR10, and IMAGENET datasets, averaged over the datasets and metrics, the algorithm is $3.8\times$ less failure-prone, and spends in total $2.5\times$ fewer queries than the current state-of-the-art attacks combined given a budget of $10,000$ queries per attack attempt. Notably, it requires no hyperparameter tuning or any data/time-dependent prior. The algorithm exploits a new approach, namely sign-based rather than magnitude-based gradient estimation. This shifts the estimation from continuous to binary black-box optimization. With three properties of the directional derivative, we examine three approaches to adversarial attacks. This yields a superior algorithm breaking a standard MNIST model using an average of 12 queries.

1. Introduction

Problem. Deep Neural Networks (DNNs) are vulnerable to adversarial examples, which are malicious inputs designed to fool the network's prediction—see (Biggio & 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 was assumed. Since the gradient points to the direction of steepest ascent, a malicious input can be perturbed along that gradient to maximize the network's loss, thereby fooling its prediction. The assumption of access to the underlying gradient does not however reflect real world scenarios. Attacks accepting a more realistic, restrictive *black-box* threat model, which do not assume access to gradients, have since been studied as will be summarized shortly.

Central to the approach of generating adversarial examples in a *black-box* threat model is estimating the gradients of the model being attacked. In estimating these gradients (their magnitudes and signs), the community at large has focused on formulating it as a problem in continuous optimization. Their works seek to reduce the query complexity from the standard $O(n)$, where n is the number of input features/covariates. In this paper, we take a different view and focus on estimating just the sign of the gradient by reformulating the problem as minimizing the Hamming distance to the gradient sign. Given access to a Hamming distance (to the gradient sign) oracle, this view guarantees a query complexity of $\Omega(n/\log_2(n+1))$: an order of magnitude lesser than the full gradient estimation's query complexity for most practically-occurring input dimensions n. Our key objective is to answer the following: *Is it possible to recover the sign of the gradient with high query efficiency and generate adversarial examples as effective as those generated by full gradient estimation approaches?*

To this end, we propose a novel formulation capitalizing on some properties of the directional derivative which, approximated by finite difference of loss queries, has been the powerhouse of black-box attacks. Particularly, this leads to the following contributions at the intersection of adversarial machine learning and black-box (zeroth-order) optimization: 1) We present three properties of the directional derivative of the loss function of the model under attack in the direction of $\{\pm 1\}^n$ vectors, and propose methods to estimate the gradient sign bits exploiting these properties. Based on one of the properties, namely separability, we devise a divide-and-conquer algorithm, which we refer to as `SignHunter`, that reduces the search complexity from 2^n sign vectors to $O(n)$. When given a budget of $O(n)$ queries, `SignHunter` is guaranteed to perform at least as well as `FGSM` (Goodfellow et al., 2015), which has access to the model's gradient. Through rigourous experiments on both standard and adversarially trained models, we

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

find that SignHunter, in its search for the gradient sign, crafts adversarial examples within a fraction of this number of queries outperforming FGSM and other state-of-the-art black-box attacks. 2) We release a software frameworkto systematically benchmark adversarial black-box attacks on DNNs for MNIST, CIFAR10, and IMAGENET datasets in terms of their success rate, query count, and other related metrics. 3) We identify several key areas of research which we believe will help the community of adversarial learning and gradient-free optimization.

Related Work. We organize the related work in two themes, namely *Adversarial Example Generation* and *Sign-Based Optimization*.

Adversarial Example Generation. This literature can be organized as generating examples in either a *white-box* or a *black-box* setting. Nelson et al. (2012) provide a theoretical framework to analyze adversarial querying in a *white-box* setting. Following the works of Biggio et al. (2013) and Goodfellow et al. (2015) who introduced the fast gradient sign method (FGSM), several methods to produce adversarial examples have been proposed for various learning tasks and threat perturbation constraints (Carlini & Wagner, 2017; Moosavi-Dezfooli et al., 2016; Hayes & Danezis, 2017; Al-Dujaili et al., 2018; Huang et al., 2018; Kurakin et al., 2017; Shamir et al., 2019). These methods assume a white-box setup and are not the focus of this work. An approach, which has received the community's attention, 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). As an alternative to the transferability phenomenon, Xiao et al. (2018) use a Generative Adversarial Network (GAN) to generate adversarial examples which are based on small norm-bounded perturbations. Both approaches involve learning on a different model, which is expensive, and does not lend itself to comparison in our setup, where we directly query the model of interest. Among works which generate examples in a *black-box* setting through iterative optimization schemes, Narodytska & Kasiviswanathan (2017) showed how a naïve policy of perturbing random segments of an image achieved adversarial example generation. They do not use any gradient information. Bhagoji et al. (2017) reduce the dimensions of the feature space using Principal Component Analysis (PCA) and random feature grouping, before estimating gradients. This enables them to bound the number of queries made. Chen et al. (2017) introduced a principled approach to solving this problem using gradient based optimization. They employ finite differences, a zeroth-order optimization tool, to estimate the gradient and then use it to design a gradient-based attack on models. While this approach successfully generates adversarial examples, it is expensive in the number of queries made to the model. 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: $\texttt{Bandits}_{TD}$. Our work contrasts the general approach used by these works. We investigate whether just estimating the *sign* of the gradient suffices to efficiently generate examples.

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 requires $\sqrt{n}$ times more iterations than signSGD, leading to a convergence rate of $O(\sqrt{n/T})$, where n is the number of optimization variables, and T is the number of iterations.

Adversarial Examples Meet Sign-based Optimization. In the context of adversarial examples generation, the effectiveness of sign of the gradient coordinates was noted in both white- and black-box settings. In the former, the Fast Gradient Sign Method (FGSM)—which is algorithmically similar to signSGD—was proposed to generate white-box adversarial examples (Goodfellow et al., 2015). Ilyas et al. (2019) examined a noisy version of FGSM to address the question of *How accurate of a gradient estimate is necessary to execute a successful attack on a neural net*. In Figure 1, we reproduce their experiment on an IMAGENET-based model—Plot (c)—and extended it to the MNIST and CIFAR10 datasets—Plots (a) and (b). Observe that estimating the sign of the top 30% gradient coordinates (in terms of their magnitudes) is enough to achieve a misclassification rate of $\sim$ 70%. Furthermore, ZO-SignSGD (Liu et al., 2019) was shown to perform better than NES at generating adversarial examples against a black-box neural network on the MNIST dataset.

2. Formal Background

Notation. Let n denote the dimension of a neural network's input. Denote a hidden n-dimensional binary code by $\boldsymbol{q}^*$. That is, $\boldsymbol{q}^* \in \mathcal{H} \equiv \{-1, +1\}^n$. The response of the Hamming (distance) oracle $\mathcal{O}$ to the ith query $\boldsymbol{q}^{(i)} \in \mathcal{H}$ is denoted by $r^{(i)} \in \{0, \dots, n\}$ and equals the Hamming distance $r^{(i)} = ||\boldsymbol{q}^{(i)} - \boldsymbol{q}^*||_H$, where the Hamming norm $||\boldsymbol{v}||_H$ is defined as the number of non-zero entries of vector $\boldsymbol{v}$. We also refer to $\mathcal{O}$ as the *noiseless* Hamming oracle,

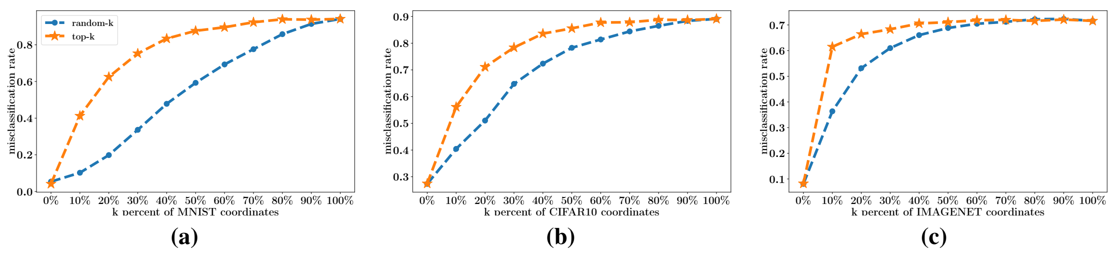


Figure 1: Misclassification rate of three neural nets (for (a) MNIST, (b) CIFAR10, and (c) IMAGENET) on the *noisy* `FGSM`'s adversarial examples as a function of correctly estimated coordinates of $\text{sign}(\nabla_{\boldsymbol{x}} f(\boldsymbol{x}, y))$ on random 1000 images from the corresponding datasets. 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 $\sim 70\%$. More details can be found in Appendix A.

in contrast to the *noisy* Hamming oracle $\hat{\mathcal{O}}$, which returns noisy versions of $\mathcal{O}$'s responses. $\mathbf{1}_n$ is the n-dimensional vector of ones. The query ratio $\rho \in (0, 1]$ is defined as m/n where m is the number of queries to $\mathcal{O}$ required to retrieve $\boldsymbol{q}^*$. Furthermore, denote the directional derivative of some function f at a point $\boldsymbol{x}$ in the direction of a vector $\boldsymbol{v}$ by $D_{\boldsymbol{v}} f(\boldsymbol{x}) \equiv \boldsymbol{v}^T \nabla_{\boldsymbol{x}} f(\boldsymbol{x})$ which often can be approximated by the *finite difference* method. That is, for $\delta > 0$, we have

$$D_{\boldsymbol{v}} f(\boldsymbol{x}) = \boldsymbol{v}^T \nabla_{\boldsymbol{x}} f(\boldsymbol{x}) \approx \frac{f(\boldsymbol{x} + \delta \boldsymbol{v}) - f(\boldsymbol{x})}{\delta} . \qquad (1)$$

Let $\Pi_S(\cdot)$ be the projection operator onto the set S, $B_p(\boldsymbol{x}, \epsilon)$ be the ℓ_p ball of radius ϵ around $\boldsymbol{x}$. Next, we provide lower and upper bounds on the query ratio ρ.

Bounds on the Query Ratio ρ. Using a packing argument, Vaishampayan (2012) proved the following lower bound on query ratio ρ.

Theorem 1. *(Vaishampayan, 2012, Theorem 1) For the noiseless Hamming oracle $\mathcal{O}$, the query ratio must satisfy $\rho = m/n \geq \frac{1}{\log_2(n+1)}$ for any sequence of m queries that determine every n-dimensional binary code $\boldsymbol{q}^*$ uniquely.*

Proof. See (Vaishampayan, 2012, Page 4). □

In the following theorem, we show that no more than n queries are required to retrieve the hidden n-dimensional binary code $\boldsymbol{q}^*$.

Theorem 2. *A hidden n-dimensional binary code $\boldsymbol{q}^* \in \mathcal{H}$ can be retrieved exactly with no more than n queries to the noiseless Hamming oracle $\mathcal{O}$.*

Proof. See Appendix C. □

3. Gradient Estimation Problem

At the heart of black-box adversarial attacks is generating a *perturbation vector* to slightly modify the original input $\boldsymbol{x}$ so as to fool the network prediction of its true label y. Put it differently, an adversarial example $\boldsymbol{x}'$ maximizes the network's loss $L(\boldsymbol{x}', y)$ but still remains ϵ_p-close to the original input $\boldsymbol{x}$. 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, to set the perturbation vector as a step in the direction of $\nabla_{\boldsymbol{x}} L(\boldsymbol{x}, y)$. Subsequently, the bulk of black-box attack methods sought to *estimate the gradient* by querying an oracle that returns, for a given input/label pair $(\boldsymbol{x}, y)$, the value of the network's loss $L(\boldsymbol{x}, y)$. 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 $(\boldsymbol{x}, y)$ in the direction of a vector $\boldsymbol{v}$, which corresponds to $\boldsymbol{v}^T \nabla_{\boldsymbol{x}} L(\boldsymbol{x}, y)$. With n linearly independent vectors $\{{\boldsymbol{v}^{(i)}}^T \nabla_{\boldsymbol{x}} L(\boldsymbol{x}, y) = d^{(i)}\}_{1 \leq i \leq n}$, 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). Moreover, the queries are not adaptive, whereas one may make use of the past queries' responses to construct the new query and recover the full gradient with less queries. Recent works tried to mitigate this issue by exploiting data- and/or time-dependent priors (Tu et al., 2018; Ilyas et al., 2018; 2019).

The lower bound of Theorem 1 on the query complexity of a Hamming oracle $\mathcal{O}$ to find a hidden vector $\boldsymbol{q}^*$ suggests the following: *instead of estimating the full gradient (sign and magnitude) and apart from exploiting any data- or time-dependent priors; why do we not focus on estimating its sign?* After all, simply leveraging (noisy) sign information of the gradient yields successful attacks; see Figure 1. Therefore, our interest in this paper is the *gradient sign estimation problem*, which we formally define next, breaking away from the general trend of the *continuous optimization view* in constructing black-box adversarial attacks, manifested by the focus on the *full gradient estimation problem*.

Definition 1. *(Gradient Sign Estimation Problem) For an input/label pair $(\boldsymbol{x}, y)$ and a loss function L, let $\boldsymbol{g}^* = \nabla_{\boldsymbol{x}} L(\boldsymbol{x}, y)$ be the gradient of L at $(\boldsymbol{x}, y)$ and $\boldsymbol{q}^* = \text{sign}(\boldsymbol{g}^*) \in \mathcal{H}$ be the sign bit vector of $\boldsymbol{g}^*$.*[1] *Then the goal*

[1]Without loss of generality, we encode the sign bit vector in

of the gradient sign estimation problem is to find a binary vector $\boldsymbol{q} \in \mathcal{H}$ *minimizing the Hamming norm*

$$\min_{\boldsymbol{q} \in \mathcal{H}} ||\boldsymbol{q} - \boldsymbol{q}^*||_H \ , \tag{2}$$

or equivalently maximizing the directional derivative[2]

$$\max_{\boldsymbol{q} \in \mathcal{H}} D_{\boldsymbol{q}} L(\boldsymbol{x}, y) \ , \tag{3}$$

from a limited number of (possibly adaptive) function value queries $L(\boldsymbol{x}', y)$.

Next, we set to tackle the problem above leveraging three properties of the loss directional derivative $D_{\boldsymbol{q}} L(\boldsymbol{x}, y)$ which, in the black-box setup, is approximated by finite difference of loss value queries $L(\boldsymbol{x}', y)$.

4. A Framework for Estimating Sign of the Gradient from Loss Oracles

Our interest is to estimate the gradient sign bits of the loss function L of the model under attack at an input/label pair $(\boldsymbol{x}, y)$ from a limited number of loss value queries $L(\boldsymbol{x}', y)$. To this end, we examine the basic concept of directional derivatives that has been employed in recent black-box adversarial attacks. Particularly, we present three approaches to estimate the gradient sign bits based on three properties of the directional derivative $D_{\boldsymbol{q}} L(\boldsymbol{x}, y)$ of the loss in the direction of a sign vector $\boldsymbol{q} \in \mathcal{H}$. For the rest of the paper, we only discuss the most successful one: Divide & Conquer. Others are described in Appendix B.

Approach 1: Divide & Conquer. Based on the definition of the directional derivative (Eq. 1), we state the following property.

Property 1 (Separability of $D_{\boldsymbol{q}} L(\boldsymbol{x}, y)$)**.** *The directional derivative* $D_{\boldsymbol{q}} L(\boldsymbol{x}, y)$ *of the loss function L at an input/label pair* $(\boldsymbol{x}, y)$ *in the direction of a binary code* $\boldsymbol{q}$ *is separable. That is,*

$$\max_{\boldsymbol{q} \in \mathcal{H}} D_{\boldsymbol{q}} L(\boldsymbol{x}, y) = \max_{\boldsymbol{q} \in \mathcal{H}} \boldsymbol{q}^T \boldsymbol{g}^* = \sum_{i=1}^{n} \max_{q_i \in \{-1,+1\}} q_i g_i^* \tag{4}$$

We employ the above property in a divide-and-conquer search which we refer to as `SignHunter`. As outlined in Algorithm 1, the technique starts with a random guess of the sign vector $\boldsymbol{q}_1$. It then proceeds to flip the sign of all the coordinates to get a new sign vector $\boldsymbol{q}_2$, and revert the flips if the loss oracle returned a value $L(\boldsymbol{x} + \delta \boldsymbol{q}_2, y)$ (or equivalently the directional derivative) less than the best obtained so far $L(\boldsymbol{x} + \delta \boldsymbol{q}_1, y)$. `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 $2^{\lceil \log(n)+1 \rceil} - 1$ sign flips to complete its search. If the query budget is not exhausted by then, one can update $\boldsymbol{x}$ with the recovered signs and restart the procedure at the updated point with a new starting code $\boldsymbol{q}_1$ ($\boldsymbol{s}$ in Algorithm 1). In the next theorem, we show that `SignHunter` is guaranteed to perform at least as well as the Fast Gradient Sign Method `FGSM` with $O(n)$ oracle queries.

Theorem 3. *(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),* `SignHunter` *is at least as effective as* `FGSM` *(Goodfellow et al., 2015) in crafting adversarial examples.*

Proof. See Appendix C. □

Theorem 3 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 fraction of this upper bound. Note that one could recover the gradient sign vector with $n + 1 < 2^{\lceil \log(n)+1 \rceil}$ queries by starting with an arbitrary sign vector and flipping its bits sequentially. Nevertheless, `SignHunter` incorporates its queries in a framework of majority voting to recover as many sign bits as possible with as few queries as possible. Consider the case where all the gradient coordinates have the same magnitude. If we start with a random sign vector whose Hamming distance to the optimal sign vector $\boldsymbol{q}^*$ is $n/2$: agreeing with $\boldsymbol{q}^*$ in the first half of coordinates. In this case, `SignHunter` needs just *three* queries to recover the entire sign vector, whereas the sequential bit flipping would require $n + 1$ queries.

Further, `SignHunter` is amenable to parallel hardware architecture and thus can carry out attacks in batches more efficiently, compared to the other two approaches we considered. We tested all the proposed approaches on a set of toy problems and found that `SignHunter` perform significantly better. For these reasons, in our experiments on the real datasets MNIST, CIFAR10, IMAGENET; we opted for `SignHunter` as our algorithm of choice to estimate the gradient sign in crafting black-box adversarial attacks as outlined in Algorithm 2.

5. Experiments

We evaluate `SignHunter` and compare it with established algorithms from the literature: `ZO-SignSGD` (Liu et al.,

$\mathcal{H} \equiv \{-1, +1\}^n$ 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 (Zhao, 2018) whose range is $\{-1, +1\}$. This follows from the observation that DNNs' gradients are not sparse (Ilyas et al., 2019, Appendix B.1).

[2]The equivalence follows from $D_{\boldsymbol{q}} L(\boldsymbol{x}, y) = \boldsymbol{q}^T \boldsymbol{g}^*$, which is maximized when $\boldsymbol{q} = \boldsymbol{q}^* = \text{sign}(\boldsymbol{g}^*)$, which in turn is a minimizer of Eq. 2.

Algorithm 1 `SignHunter`

Input:

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

```
def init(g) :
    $i \leftarrow 0$
    $h \leftarrow 0$
    $g \leftarrow g$
    $\boldsymbol{s} \sim \mathcal{U}(\mathcal{H})$        // e.g., all ones $[+1, +1, \ldots, +1]$
    done $\leftarrow false$
    $g_{best} \leftarrow -\infty$

def is_done() :
    return done

def step() :
    chunk_len $\leftarrow \lceil n/2^h \rceil$
    flip the bits of $\boldsymbol{s}$ indexed from $i*$chunk_len till
      $(i+1)*$chunk_len
    if $g(\boldsymbol{s}) \geq g_{best}$:
        $g_{best} \leftarrow g(\boldsymbol{s})$
    else:
        flip back the bits of $\boldsymbol{s}$ indexed from
          $i*$chunk_len till $(i+1)*$chunk_len
    increment $i$
    if $i == 2^h$:
        $i \leftarrow 0$
        increment $h$
        if $h == \lceil \log_2(n) \rceil + 1$: done $\leftarrow true$

def get_current_sign_estimate() :
    return $\boldsymbol{s}$
```

Algorithm 2 Black-Box Adversarial Example Generation with `SignHunter`

Input:

$\boldsymbol{x}_{init}$	: input to be perturbed,
y_{init}	: $\boldsymbol{x}_{init}$'s true label,
$B_p(., \epsilon)$	: ℓ_p perturbation ball of radius ϵ
L	: loss function of the neural net under attack

```
1:  $\delta \leftarrow \epsilon$    // set finite-difference probe to perturbation bound
2:  $\boldsymbol{x}_o \leftarrow \boldsymbol{x}_{init}$
3:  Define the function $g$ as
      $g(\boldsymbol{q}) = \frac{L(\Pi_{B_p(\boldsymbol{x}_{init},\epsilon)}(\boldsymbol{x}_o + \delta\boldsymbol{q}), y_{init}) - L(\boldsymbol{x}_o, y_{init})}{\delta}$
4:  SignHunter.init($g$)
5:  //$C(\cdot)$ returns top class
6:  while $C(\boldsymbol{x}) = y_{init}$ do
7:    SignHunter.step()
8:    $\boldsymbol{s} \leftarrow$ SignHunter.get_current_sign_estimate()
9:    $\boldsymbol{x} \leftarrow \Pi_{B_p(\boldsymbol{x}_{init},\epsilon)}(\boldsymbol{x}_o + \delta\boldsymbol{s})$
10:   if SignHunter.is_done() then
11:     $\boldsymbol{x}_o \leftarrow \boldsymbol{x}$
12:     Define the function $g$ as
          $g(\boldsymbol{q}) = \frac{L(\Pi_{B_p(\boldsymbol{x}_{init},\epsilon)}(\boldsymbol{x}_o + \delta\boldsymbol{q}), y_{init}) - L(\boldsymbol{x}_o, y_{init})}{\delta}$
13:     SignHunter.init($g$)
14:   end if
15: end while
16: return $\boldsymbol{x}$
```

2019), `NES` (Ilyas et al., 2018), and `Bandits`$_{TD}$ (Ilyas et al., 2019) in terms of their effectiveness in crafting (without loss of generality) untargeted black-box adversarial examples. Both ℓ_∞ and ℓ_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 of setting the perturbation bound ϵ, arbitrary bounds were used in several papers. We set the perturbation bounds based on the following.

For the ℓ_∞ 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 ℓ_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. For MNIST and CIFAR10, the naturally trained models from (Madry et al., 2017)'s MNIST[3] and CIFAR10[4] challenges are used. For IMAGENET, the *Inception-V3* model from TensorFlow is used.[5] The loss oracle represents the cross-entropy loss of the respective model. General setup of the experiments is summarized in Appendix D.

Hyperparameters 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. This can be attributed to either using a different software framework, models, or the way the model's inputs are transformed. We made use of a synthetic concave loss function to 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`. Further, `Bandits`$_{TD}$ outperformed the rest of the algorithms towards the end of query budget. That

[3] https://github.com/MadryLab/mnist_challenge
[4] https://github.com/MadryLab/cifar10_challenge
[5] https://bit.ly/2VYDc4X

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 δ for `SignHunter` is set to the perturbation bound ϵ because this perturbation is used for for both computing the finite difference and crafting the adversarial examples—see Line 1 in Algorithm 2. 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 as discussed above. More details on the hyperparameters setup can be found in Appendix D.

Results. Figure 2 shows the trade-off between the success (evasion) rate and the mean number of queries (of the successful attacks) needed to generate an adversarial example for the MNIST, CIFAR10, and IMAGENET classifiers in the ℓ_∞ and ℓ_2 perturbation constraints. In other words, these figures indicate the average number of queries required for a desired success rate. Tabulated summary of these plots can be found in Appendix E, namely Tables 6, 7, and 8. Furthermore, we plot the classifier 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 used in Figures 10, 11, and 12 of Appendix E. Based on the results, we observe the following:

For any given success rate, `SignHunter` dominates the previous state of the art approaches in all settings except the IMAGENET ℓ_2 setup,[6] where $\texttt{Bandits}_{TD}$ shows a better query efficiency when the desired success rate is greater than or equal ~ 0.35. Our approach is remarkably efficient in the ℓ_∞ setup (e.g., achieving a $\mathbf{100\%}$ evasion using—on average—just $\mathbf{12}$ queries per image against the MNIST classifier!). Its performance degrades—yet, still outperforms the rest, most of the time—in the ℓ_2 setup. This is expected, since `SignHunter` perturbs all the coordinates with the same magnitude and the ℓ_2 perturbation bound ϵ_2 for all the datasets in our experiments is set such that $\epsilon_2/\sqrt{n} \ll \epsilon_\infty$ as shown in Table 1 of Appendix D. Take the case of MNIST ($n = 28 \times 28$), where $\epsilon_\infty = 0.3$ and $\epsilon_2 = 3$. For `SignHunter`, the ℓ_2 setup is equivalent to an ℓ_∞ perturbation bound of $3/28 \approx 0.1$. The employed ℓ_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 2.1 each; ten pixels by a magnitude of 0.9 each, etc. On the other hand, the binary optimization view of `SignHunter` limits it to always perturb all 28×28 pixels by a magnitude of 0.1. Despite its less degrees of freedom, `SignHunter` maintains its effectiveness in the ℓ_2 setup. The plots can be viewed as a sensitivity assessment of `SignHunter` as ϵ gets smaller for each dataset. Moreover, the performance of `SignHunter` is in line with Theorem 3 when compared with the performance of `FGSM` (the noisy `FGSM` at $k = 100\%$ in Figures 1 and 2 of Appendix A) in both ℓ_∞ and ℓ_2 setups for MNIST and CIFAR10—for IMAGENET, $2n = 536,406$ is beyond our query budget of $10,000$ queries. For example, `FGSM` has a failure rate of 0.32 for CIFAR10 ℓ_2 (Appendix A, Figure 2 (b)), while `SignHunter` achieves a failure rate of 0.21 with $692.39 < 2n = 2 \times 3 \times 32 \times 32 = 6144$ queries (Appendix E, Table 7).

[6] Strictly speaking, all the algorithms are comparable in the CIFAR10 ℓ_2 setup for success rate ≤ 0.3.

Table 1: Top-3 attacks on the MNIST black-box challenge. Adapted from the challenge's website—as of Feb 22, 2019.

Black-Box Attack	Model Accuracy
`SignHunter` (Algorithm 2)	**91.47%**
(Xiao et al., 2018)	92.76%
`PGD` against three independently and adversarially trained copies of the network	93.54%

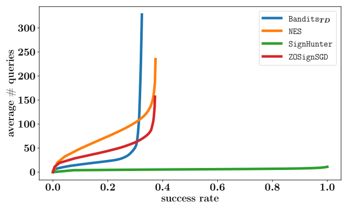
Incorporating `SignHunter` in an iterative framework of perturbing the data point $\boldsymbol{x}$ till the query budget is exhausted (Lines 10 to 14 in Algorithm 2) supports the observation in white-box settings that iterative `FGSM`—or `Projected Gradient Descent` (`PGD`)—is stronger than `FGSM` (Madry et al., 2017; Al-Dujaili et al., 2018). This is evident by the upticks in `SignHunter`'s performance on the MNIST ℓ_2 case (Figure 10 of Appendix E: classifier's loss, Cosine and Hamming similarity plots), which happens after every iteration (after every other $2 \times 28 \times 28$ queries). Plots of the Hamming similarity capture the quality of the gradient sign estimation in terms of Eq. 2, while plots of the average Cosine similarity capture it in terms of Eq. 3. Both `SignHunter` and $\texttt{Bandits}_{TD}$ consistently optimize both objectives. In general, `SignHunter` enjoys a faster convergence especially on the Hamming metric as it is estimating the signs compared to $\texttt{Bandits}_{TD}$'s full gradient estimation. This is highlighted in the IMAGENET ℓ_2 setup. Note that once an attack is successful, the gradient sign estimation at that point is used for the rest of the plot. This explains why, in the ℓ_∞ settings, `SignHunter`'s plot does not improve compared to its ℓ_2 counterpart, as most of the attacks are successful in the very first few queries made to the loss oracle.

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. The number of queries spent is computed based on Tables 6, 7, and 8 of Appendix E as

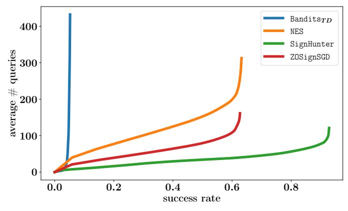
*(1 - fail_rate) * avg_#_queries + fail_rate * 10,000.*

6. Attack Effectiveness Under Defenses

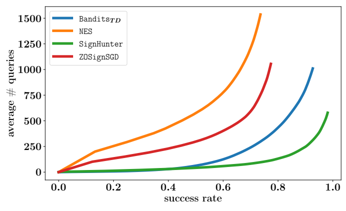
To complement our results in Section 5, we evaluated `SignHunter` against *adversarial training*, an effective



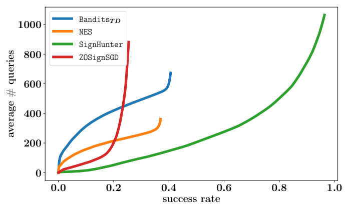
(a) MNIST ℓ_∞



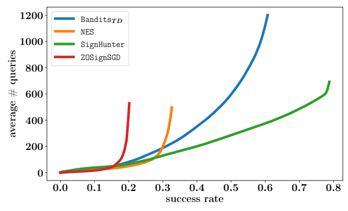
(b) CIFAR10 ℓ_∞



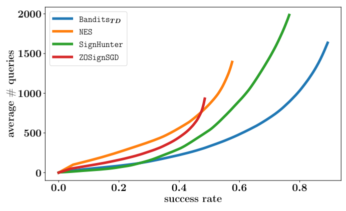
(c) IMAGENET ℓ_∞



(d) MNIST ℓ_2



(e) CIFAR10 ℓ_2



(f) IMAGENET ℓ_2

Figure 2: Performance of black-box attacks in the ℓ_∞ and ℓ_2 perturbation constraint. The plots show the average number of queries used per successful image for each attack when reaching a specified success rate.

Table 2: Top-3 attacks for the CIFAR10 black-box challenge. Adapted from the challenge's website—as of Feb 22, 2019.

Black-Box Attack	**Model Accuracy**
`SignHunter` (Algorithm 2)	**47.16%**
`PGD` on the cross-entropy loss for the adversarially trained public network	63.39%
`PGD` on the CW loss for the adversarially trained public network	64.38%

Table 3: 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., 2017, Table 4).

Model	**clean**	**Max. Black-box**	**`SignHunter`** after 20 queries	**`SignHunter`** after 1000 queries
v3$_{\text{adv-ens4}}$	24.2 (26.73)	33.4	(40.61)	**(90.75)**

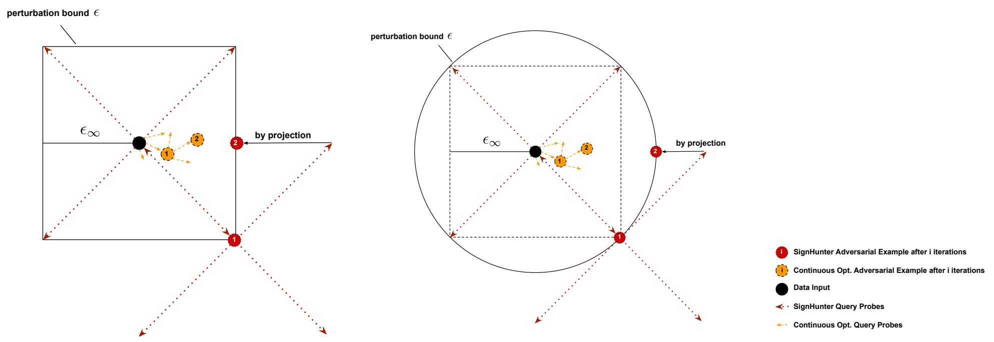
way to improve the robustness of DNNs (Madry et al., 2017). In particular, we attacked the *secret* models used in public challenges for MNIST and CIFAR10. There was no corresponding challenge for IMAGENET. Instead, we used *ensemble adversarial training*, a method that argues security against black-box attacks based on transferability/substitute models (Tramèr et al., 2017). The same metrics used in Section 5 are recorded for the experiments here in Appendix F.

Public MNIST Black-Box Attack Challenge. In line with the challenge setup, $10,000$ test images were used with an ℓ_∞ 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 5. No maximum query budget was specified, so we set it to $5,000$ queries. This is similar to the number of iterations given to a `PGD` attack in the white-box setup of the challenge: 100-steps with 50 random restarts. As shown in Table 1, `SignHunter`'s attacks resulted in the lowest model accuracy of $\mathbf{91.47}\%$, outperforming all other state-of-the-art black-box attack strategies submitted to the challenge with an average number of queries of $\mathbf{233}$ per successful attack. Note that the most powerful *white-box* attack by Zheng et al. (2018)—as of Feb 22, 2019—resulted in a model accuracy of 88.56%–not shown in the table.

Public CIFAR10 Black-Box Attack Challenge. In line with the challenge setup, $10,000$ test images were used with an ℓ_∞ perturbation bound of $\epsilon = 8$. Although the secret model is released, we treated it as a black box similar to our experiments in Section 5. Similar to the MNIST challenge, the query budget is $5,000$ queries. From Table 2, `SignHunter`'s attacks resulted in the lowest model accuracy of $\mathbf{47.16}\%$, outperforming all other state-of-the-art black-box attack strategies submitted to the challenge with an average number of queries of **569** per successful attack. Note that the most powerful *white-box* attack by Zheng et al. (2018), —as of Feb 22, 2019—resulted in a model accuracy of 44.71%–not shown in the table.

Ensemble Adversarial Training on IMAGENET. In line with (Tramèr et al., 2017), we set $\epsilon = 0.0625$ and report the model's misclassification over 10,000 random images from IMAGENET's validation set. We attack the v3$_{\text{adv-ens4}}$ model.[7] As shown in Table 3, 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%!

[7] https://bit.ly/2XWTdKx


perturbation bound ϵ
ϵ_∞
by projection
perturbation bound ϵ
ϵ_∞
by projection
SignHunter Adversarial Example after i iterations
Continuous Opt. Adversarial Example after i iterations
Data Input
SignHunter Query Probes
Continuous Opt. Query Probes

(a) ℓ_∞ perturbation (b) ℓ_2 perturbation

Figure 3: Illustration of adversarial examples crafted by `SignHunter` in comparison to attacks that are based on the continuous optimization in both (a) ℓ_∞ and (b) ℓ_2 settings. If `SignHunter` is given a query budget $> 2n$, which is the case here, the crafted adversarial examples are not necessary at the perturbation vertices, e.g., the red ball 2.

7. Open Questions

Priors. Current version of `SignHunter` does not exploit any data- or time-dependent priors. With these priors, algorithms such as `Bandits`$_{TD}$ operate on a search space of dimensionality $\sim 36\times$ less than that of `SignHunter` for IMAGENET. *Can such priors be incorporated in `SignHunter` to have a data-dependent grouping of gradient coordinates instead of the current equal-size grouping?*

Adversarial Training. Compared to other attacks that are based on transferability and generative adversarial networks, our approach showed more effectiveness towards (ensemble) adversarial training. Standard adversarial training relies on attacks that employ iterative continuous optimization methods such as `PGD` in contrast to our attack which stems from a binary optimization view. *What are the implications?*

Perturbation Vertices.[8] Using its first $O(n)$ queries, `SignHunter` probes $O(n)$ extreme points of the perturbation region as potential adversarial examples, while iterative continuous optimization such as `NES` probes points in the Gaussian sphere around the current point as shown in Figure 3. *Does looking up extreme points (vertices) of the perturbation region suffice to craft adversarial examples? If that is the case, how to efficiently search through them?* `SignHunter` searches through $2n$ vertices out of 2^n and it could find adversarial examples among a tiny fraction of these vertices. Recall, in the MNIST ℓ_∞ setup in Section 5, it was enough to look up just $\sim$ 12 out of 2^{784} vertices for each image achieving a 100% evasion over $1,000$ images. Note that after $2n$ queries, `SignHunter` may not visit other vertices as they will be $2\epsilon_\infty$ away as shown in Figure 3. We ignored this effect in our experiments.[9] *Will `SignHunter` be more effective if the probes are made strictly at the perturbation vertices?* This question shows up clearly in the public MNIST challenge where the loss value at the potential adversarial examples dips after every $\sim 2n$ queries (see top left plot of Figure 13 in Appendix!F). We conjecture the reason is that these potential adversarial examples are not extreme points as illustrated in Figure 3: they are like the red ball 2 rather than the red ball 1.

[8]We define perturbation vertices as extreme points of the region $B_p(\boldsymbol{x}, \epsilon)$. That is, $\boldsymbol{x} \pm \epsilon_\infty$, where $\epsilon_\infty = \epsilon$ when $p = \infty$ and $\epsilon_\infty = \epsilon/\sqrt{n}$ when $p = 2$. See Figure 3.

[9]This effect is negligible for IMAGENET as $2n < 10,000$.

8. 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: minimizing the Hamming distance to the gradient sign or, equivalently, maximizing the directional derivative. Approximated by the finite difference of the loss value queries, we examine three properties of the directional derivative of the model's loss in the direction of $\{\pm 1\}^n$ vectors. The separability property helped us devise `SignHunter`, a divide-and-conquer algorithm that is guaranteed to perform *at least* as well as `FGSM` after $O(n)$ queries. In practice, `SignHunter` needs a fraction of this number of queries to craft adversarial examples. To verify its effectiveness on real-world datasets, `SignHunter` was evaluated on neural network models for the MNIST, CIFAR10, and IMAGENET datasets. `SignHunter` yields black-box attacks that are $2.5\times$ more query efficient and $3.8\times$ less failure-prone than the state of the art attacks combined. Moreover, `SignHunter` achieves the highest evasion rate on two public black-box attack challenges. We also show that models that are robust against substitute-model attacks are vulnerable to our attack. Our future work will investigate several research questions.

References

Al-Dujaili, A., Huang, A., Hemberg, E., and O'Reilly, U.-M. Adversarial deep learning for robust detection of binary encoded malware. In *2018 IEEE Security and Privacy Workshops (SPW)*, pp. 76–82. IEEE, 2018.

Bernstein, J., Wang, Y.-X., Azizzadenesheli, K., and Anandkumar, A. signSGD: Compressed optimisation for non-convex problems. In Dy, J. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 560–569, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR. URL http://proceedings.mlr.press/v80/bernstein18a.html.

Bhagoji, A. N., He, W., Li, B., and Song, D. Exploring the space of black-box attacks on deep neural networks. *arXiv preprint arXiv:1712.09491*, 2017.

Biggio, B. and Roli, F. Wild patterns: Ten years after the rise of adversarial machine learning. *Pattern Recognition*, 84:317–331, 2018.

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 *Joint European conference on machine learning and knowledge discovery in databases*, pp. 387–402. Springer, 2013.

Carlini, N. and Wagner, D. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy (SP)*, pp. 39–57. IEEE, 2017.

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. ACM, 2017.

Cohen, J., Rosenfeld, E., and Kolter, J. Z. Certified adversarial robustness via randomized smoothing. *arXiv:1902.02918v1*, 2019.

Goodfellow, I., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*, 2015. URL http://arxiv.org/abs/1412.6572.

Hayes, J. and Danezis, G. Machine learning as an adversarial service: Learning black-box adversarial examples. *CoRR*, abs/1708.05207, 2017.

Huang, A., Al-Dujaili, A., Hemberg, E., and O'Reilly, U.-M. On visual hallmarks of robustness to adversarial malware. *arXiv preprint arXiv:1805.03553*, 2018.

Ilyas, A., Engstrom, L., Athalye, A., and Lin, J. Black-box adversarial attacks with limited queries and information. In Dy, J. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 2137–2146, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR. URL http://proceedings.mlr.press/v80/ilyas18a.html.

Ilyas, A., Engstrom, L., and Madry, A. Prior convictions: Black-box adversarial attacks with bandits and priors. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=BkMiWhR5K7.

Kurakin, A., Goodfellow, I. J., and Bengio, S. Adversarial machine learning at scale. 2017. URL https://arxiv.org/abs/1611.01236.

Liu, S., Chen, P.-Y., Chen, X., and Hong, M. signSGD via zeroth-order oracle. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=BJe-DsC5Fm.

Liu, Y., Chen, X., Liu, C., and Song, D. Delving into transferable adversarial examples and black-box attacks. *arXiv preprint arXiv:1611.02770*, 2016.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

Maurer, P. M. A search strategy using a hamming-distance oracle. 2009.

Moosavi-Dezfooli, S.-M., Fawzi, A., and Frossard, P. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2574–2582, 2016.

Narodytska, N. and Kasiviswanathan, S. P. Simple black-box adversarial attacks on deep neural networks. In *CVPR Workshops*, volume 2, 2017.

Nelson, B., Rubinstein, B. I., Huang, L., Joseph, A. D., Lee, S. J., Rao, S., and Tygar, J. Query strategies for evading convex-inducing classifiers. *Journal of Machine Learning Research*, 13(May):1293–1332, 2012.

Papernot, N., McDaniel, P., Goodfellow, I., Jha, S., Celik, Z. B., and Swami, A. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, pp. 506–519. ACM, 2017.

Shamir, A., Safran, I., Ronen, E., and Dunkelman, O. A simple explanation for the existence of adversarial examples with small hamming distance. *arXiv preprint arXiv:1901.10861*, 2019.

Tramèr, F., Kurakin, A., Papernot, N., Goodfellow, I., Boneh, D., and McDaniel, P. Ensemble adversarial training: Attacks and defenses. *arXiv preprint arXiv:1705.07204*, 2017.

Tu, C.-C., Ting, P., Chen, P.-Y., Liu, S., Zhang, H., Yi, J., Hsieh, C.-J., and Cheng, S.-M. Autozoom: Autoencoder-based zeroth order optimization method for attacking black-box neural networks. *arXiv preprint arXiv:1805.11770*, 2018.

Vaishampayan, V. A. Query matrices for retrieving binary vectors based on the hamming distance oracle. *arXiv preprint arXiv:1202.2794*, 2012.

Xiao, C., Li, B., Zhu, J.-Y., He, W., Liu, M., and Song, D. Generating adversarial examples with adversarial networks, 2018. URL https://openreview.net/forum?id=HknbyQbC-.

Zhao, Y.-B. *Sparse optimization theory and methods*. CRC Press, an imprint of Taylor and Francis, Boca Raton, FL, 2018. ISBN 978-1138080942.

Zheng, T., Chen, C., and Ren, K. Distributionally adversarial attack. *arXiv preprint arXiv:1808.05537*, 2018.