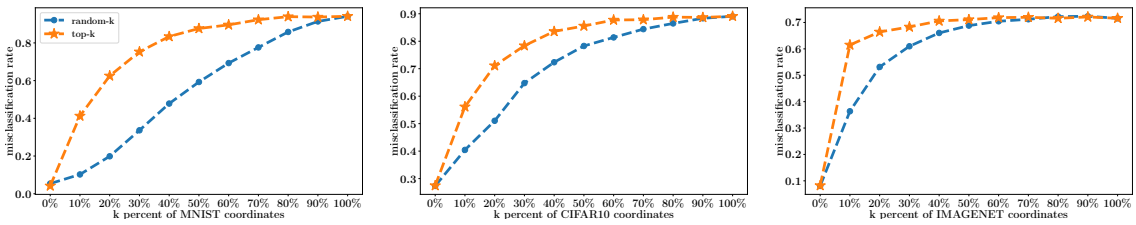
ICML 2019–SUPPLEMENTARY MATERIAL
There are No Bit Parts for Sign Bits in Black-Box Attacks

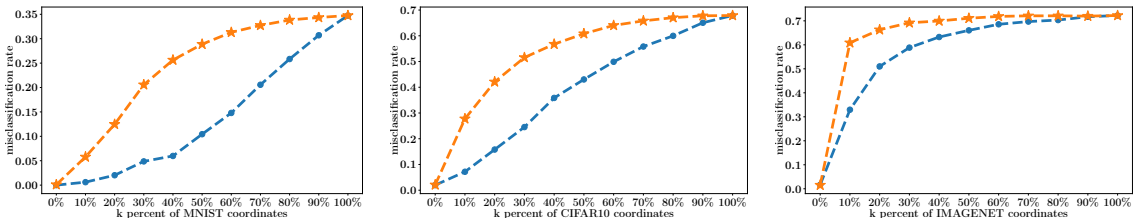
Appendix A. Noisy `FGSM`

This section shows the performance of the noisy `FGSM` on standard models (described in Section 5 of the main paper) on the MNIST, CIFAR10 and IMAGENET datasets. In Figure 1, we consider the ℓ_∞ threat perturbation constraint. Figure 2 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 $|\partial L(\boldsymbol{x}, y)/\partial x_i|$ (`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 when compared to `SignHunter`'s performance shown in Appendix C.



(a) **(b)** **(c)**

Figure 1. 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 $\text{sign}(\nabla_{\boldsymbol{x}} f(\boldsymbol{x}, y))$ on random 1000 images from the corresponding evaluation dataset, with the maximum allowed ℓ_∞ perturbation ϵ 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.



(a) **(b)** **(c)**

Figure 2. 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 $\text{sign}(\nabla_{\boldsymbol{x}} f(\boldsymbol{x}, y))$ on random 1000 images from the corresponding evaluation dataset, with the maximum allowed ℓ_2 perturbation ϵ being set to 3, 127, and 5, respectively. Compared to Figure 1, the performance on MNIST and CIFAR10 drops significantly.

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

Our interest in this section 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 $\boldsymbol{q} \in \mathcal{H}$.

Approach 1: Divide & Conquer

Covered in the main article.

Approach 2: Loss Oracle as a Noisy Hamming Oracle

The directional derivative of the loss function L at $(\boldsymbol{x}, y)$ in the direction of a binary code $\boldsymbol{q}$ can be written as

$$\begin{aligned} D_{\boldsymbol{q}}L(\boldsymbol{x}, y) &= \boldsymbol{q}^T \boldsymbol{g}^* \\ &= \sum_{i \in \mathcal{I}_{\boldsymbol{q}}^+} |g_i^*| - \sum_{i \in \mathcal{I}_{\boldsymbol{q}}^-} |g_i^*| \\ &= |\mathcal{I}_q^+| \bar{g}_{\mathcal{I}_{\boldsymbol{q}}^+} - |\mathcal{I}_q^-| \bar{g}_{\mathcal{I}_{\boldsymbol{q}}^-} , \end{aligned} \tag{1}$$

where $\mathcal{I}_{\boldsymbol{q}}^+ \equiv \{i \mid i \in [n] \ , \ q_i^* = q_i\}$, $\mathcal{I}_{\boldsymbol{q}}^- \equiv [n] \setminus \mathcal{I}_{\boldsymbol{q}}^+$. Note that $|\mathcal{I}_q^+| + |\mathcal{I}_q^-| = n$. The quantities $\bar{g}_{\mathcal{I}_{\boldsymbol{q}}^+}$ and $\bar{g}_{\mathcal{I}_{\boldsymbol{q}}^-}$ are the means of $\{|g_i|\}_{i \in \mathcal{I}_{\boldsymbol{q}}^+}$ and $\{|g_i|\}_{i \in \mathcal{I}_{\boldsymbol{q}}^-}$, respectively. Observe that $|\mathcal{I}_{\boldsymbol{q}}^-| = ||\boldsymbol{q} - \boldsymbol{q}^*||_H$: the Hamming distance between $\boldsymbol{q}$ and the gradient sign $\boldsymbol{q}^*$. In other words, the directional derivative $D_{\boldsymbol{q}}L(\boldsymbol{x}, y)$ has the following property.

Property 2. *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 $\boldsymbol{q}$ can be written as an affine transformation of the Hamming distance between $\boldsymbol{q}$ and $\boldsymbol{q}^*$. Formally, we have*

$$D_{\boldsymbol{q}}L(\boldsymbol{x}, y) = n\bar{g}_{\mathcal{I}_{\boldsymbol{q}}^+} - (\bar{g}_{\mathcal{I}_{\boldsymbol{q}}^-} + \bar{g}_{\mathcal{I}_{\boldsymbol{q}}^+})||\boldsymbol{q} - \boldsymbol{q}^*||_H \tag{2}$$

If we can recover the Hamming distance from the directional derivative based on Eq. 2, efficient Hamming search strategies—e.g., (Maurer, 2009)—can then be used to recover the gradient sign bits $\boldsymbol{q}^*$ with a query complexity $\Omega(n/\log_2(n+1))$ as stated in Theorem 1. However, not all terms of Eq. 2 is known to us. While n is the number of data features (known a priori) and $D_{\boldsymbol{q}}L(\boldsymbol{x}, y)$ is available through a finite difference oracle, $\bar{g}_{\mathcal{I}_{\boldsymbol{q}}^+}$ and $\bar{g}_{\mathcal{I}_{\boldsymbol{q}}^-}$ are not known. Here, we propose to approximate these values by their Monte Carlo estimates: averages of the magnitude of sampled gradient components. Our assumption is that the magnitudes of gradient coordinates are not very different from each other, and hence a Monte Carlo estimate is good enough (with small variance). Our experiments on MNIST, CIFAR10, and IMAGENET confirm the same—see Figure 14 in the supplement.

To use the ith gradient component g_i^* as a sample for our estimation, one can construct two binary codes $\boldsymbol{u}$ and $\boldsymbol{v}$ such that *only* their ith bit is different, i.e., $||\boldsymbol{u} - \boldsymbol{v}||_H = 1$. Thus, we have

$$|g_i^*| = \frac{|D_{\boldsymbol{u}}L(\boldsymbol{x}, y) - D_{\boldsymbol{v}}L(\boldsymbol{x}, y)|}{2} \tag{3}$$

$$q_i^* = \text{sign}(g_i^*) = \begin{cases} u_i & \text{if } D_{\boldsymbol{u}}L(\boldsymbol{x}, y) > D_{\boldsymbol{v}}L(\boldsymbol{x}, y) \\ v_i & \text{otherwise .} \end{cases} \tag{4}$$

Let $\mathcal{D}$ be the set of indices of gradient components we have recovered—magnitude and sign—so far through Eq. 3 and Eq. 4. Then,

$$\bar{g}_{\mathcal{I}_{\boldsymbol{q}}^+} \approx \frac{1}{|\mathcal{D}_{\boldsymbol{q}}^+|} \sum_{d \in \mathcal{D}_{\boldsymbol{q}}^+} |g_d^*| \ , \tag{5}$$

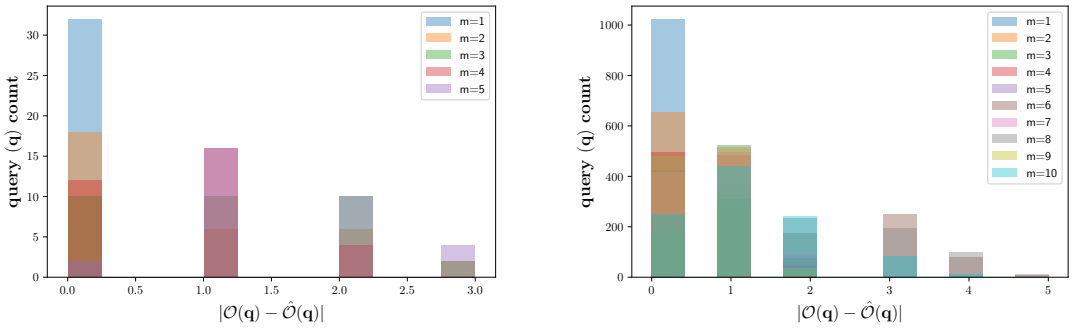
$$\bar{g}_{\mathcal{I}_{\boldsymbol{q}}^-} \approx \frac{1}{|\mathcal{D}_{\boldsymbol{q}}^-|} \sum_{d \in \mathcal{D}_{\boldsymbol{q}}^-} |g_d^*| \ , \tag{6}$$

where $\mathcal{D}_{\boldsymbol{q}}^+ \equiv \{d \mid d \in \mathcal{D} \ , \ q_d^* = q_i\}$ and $\mathcal{D}_{\boldsymbol{q}}^- \equiv \mathcal{D} \setminus \mathcal{D}_{\boldsymbol{q}}^+$.[1] As a result, the Hamming distance between $\boldsymbol{q}$ and the gradient sign $\boldsymbol{q}^*$ can be approximated with the following quantity, which we refer to as the *noisy* Hamming oracle $\hat{\mathcal{O}}$.

$$||\boldsymbol{q} - \boldsymbol{q}^*||_H \approx \frac{\frac{n}{|\mathcal{D}_{\boldsymbol{q}}^+|} \sum_{d \in \mathcal{D}_{\boldsymbol{q}}^+} |g_d^*| - D_{\boldsymbol{q}}L(\boldsymbol{x}, y)}{\frac{1}{|\mathcal{D}_{\boldsymbol{q}}^+|} \sum_{d \in \mathcal{D}_{\boldsymbol{q}}^+} |g_d^*| + \frac{1}{|\mathcal{D}_{\boldsymbol{q}}^-|} \sum_{d \in \mathcal{D}_{\boldsymbol{q}}^-} |g_d^*|} \tag{7}$$

We empirically evaluated the quality of $\hat{\mathcal{O}}$'s responses on a toy problem where we controlled the magnitude spread/concentration of the gradient coordinates with m being the number of unique values (magnitudes) of the gradient coordinates. As detailed in Figure 3, the error can reach $\lceil n/2 \rceil$. This a big mismatch, especially if we recall the Hamming distance's range is $[0, n]$. The negative impact of this on the Hamming search strategy by Maurer (2009) was verified empirically in Figure 4. We considered the simplest case where `Maurer's` was given access to the noisy Hamming oracle $\hat{\mathcal{O}}$ in a setup similar to the one outlined in Figure 3, with $n = 80$, $|\mathcal{D}| = n/4 = 20$, $m \in \{1, 2\}$, and the hidden code $\boldsymbol{q}^* = [+1, \ldots, +1]$. To account for the randomness in constructing $\mathcal{D}$, we ran 30 independent runs and plot the average Hamming distance (with confidence bounds) over `Maurer's` queries. In Figure 4 (a), $m = 1$ which corresponds to exact estimation $\hat{\mathcal{O}} = \mathcal{O}$, `Maurer's` spends 21 queries to construct $\mathcal{D}$ and terminates one query

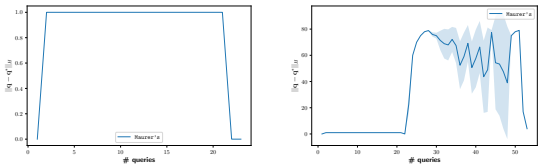
[1] It is possible that one of $\mathcal{D}_{\boldsymbol{q}}^+$ and $\mathcal{D}_{\boldsymbol{q}}^-$ will $\emptyset$ (e.g., when we only have one sample). In this case, we make the approximation as $\bar{g}_{\mathcal{I}_{\boldsymbol{q}}^+} = \bar{g}_{\mathcal{I}_{\boldsymbol{q}}^-} \approx \frac{1}{|\mathcal{D}|} \sum_{d \in \mathcal{D}} |g_d^*|$.



(a) $n = 5$ (b) $n = 10$

Figure 3. The error distribution of the *noisy* Hamming oracle $\hat{\mathcal{O}}$ (right side of Eq. 7) compared to the *noiseless* counterpart $\mathcal{O}$ (left side of Eq. 7) as a function of the number of unique values (magnitudes) of the gradient coordinates m. Here, $L(\boldsymbol{x}, y)$ has the form $\mathbf{c}^T\boldsymbol{x}$. That is, $m = |\texttt{uniq}(|\mathbf{c}|)| \leq n$ with $n \in \{5, 10\}$ being the input length. With $m = 1$, the estimation is exact ($\hat{\mathcal{O}} = \mathcal{O}$) for all the binary code queries $\boldsymbol{q}$—32 codes for $n = 5$, 1028 codes for $n = 10$. The error seems to be bounded by $\lceil \frac{n}{2} \rceil$. For a given m, c_i—the i^{th} coordinate of $\mathbf{c}$—is randomly assigned a value from the m evenly spaced numbers in the range $[0.1, m/n]$. We set the size of the sampled gradient coordinates set $|\mathcal{D}|$ to $\lfloor n/4 \rfloor$.

afterwards with the true binary code $\boldsymbol{q}^*$, achieving a query ratio of 21/80. On the other hand, when we set $m = 2$ in Figure 4 (b); `Maurer's` returns a 4-Hamming-distance away solution from the true binary code $\boldsymbol{q}^*$ after 51 queries. This is not bad for an 80-bit long code. However, this is in a tightly controlled setup where the gradient magnitudes are just one of two values. To be studied further is the bias/variance decomposition of the returned solution and the corresponding query ratio. We leave this investigation for future work.



(a) $m = 1$ (b) $m = 2$

Figure 4. Performance of `Maurer's` on the noisy Hamming oracle $\hat{\mathcal{O}}$. The setup is similar to that of Figure 3, with $n = 80$ and $m \in \{1, 2\}$.

Approach 3: Optimism in the Face of Uncertainty

In the previous approach, we considered the approximated Hamming distance (Eq. 7) as a surrogate for the formal optimization objective (Eq. 3 in the main paper) of the gradient sign estimation problem. We found that current Hamming search strategies are not robust to approximation error. In this approach, we consider maximizing the directional derivative (Eq. 4 in the main paper)

$$\max_{\boldsymbol{q} \in \mathcal{H}} D_{\boldsymbol{q}} L(\boldsymbol{x}, y) \ , \tag{8}$$

as our formal objective of the gradient sign estimation problem. Formally, we treat the problem as a binary black-box optimization over the 2^n hypercube vertices, which correspond to all possible sign vectors. This is significantly worse than $O(n)$ of the continuous optimization view. Nevertheless, the rationale here is that we do not need to solve Eq. 8 to optimality (recall the effectiveness of noisy `FGSM` in Appendix A); we rather need a fast convergence to a suboptimal but *adversarially helpful* sign vector $\boldsymbol{q}$. In addition, the continuous optimization view often employs an iterative scheme of T steps within the perturbation ball $B_p(\boldsymbol{x}, \epsilon)$, calling the *gradient estimation routine* in every step leading to a search complexity of nT. In our setup, we use the best obtained solution for Eq. 8 so far in a similar fashion to the noisy `FGSM`. In other words, our *gradient sign estimation routine* runs at the top level of our adversarial example generation procedure instead of calling it as a subroutine. In this and the next approach, we address the following question: *how do we solve Eq. 8?*

Optimistic methods, i.e., methods that implement the optimism in the face of uncertainty principle have demonstrated a theoretical as well as empirical success when applied to black-box optimization problems (Munos, 2011; Al-Dujaili & Suresh, 2017; 2018). Such a principle finds its foundations in the machine learning field addressing the exploration vs. exploitation dilemma, known as the multi-armed bandit problem. Within the context of function optimization, optimistic approaches formulate the complex problem of optimizing an arbitrary black-box function g (e.g., Eq. 8) over the search space ($\mathcal{H}$ in this paper) as a hierarchy of simple bandit problems (Kocsis & Szepesvári, 2006) in the form of space-partitioning tree search $\mathcal{T}$. At step t, the al-

gorithm optimistically expands a leaf node (partitions the corresponding subspace) from the set of leaf nodes $\mathcal{L}_t$ that may contain the global optimum. The i^{th} node at depth h, denoted by (h, i), corresponds to the subspace/cell $\mathcal{H}_{h,i}$ such that $\mathcal{H} = \cup_{0 \le i < K^h} \mathcal{H}_{h,i}$. To each node (h, i), a representative point $\boldsymbol{q}_{h,i} \in \mathcal{H}_{h,i}$ is assigned, and the value of the node (h, i) is set to $g(\boldsymbol{q}_{h,i})$. See Figure 7 for an example of a space-partitioning tree $\mathcal{T}$ of $\mathcal{H}$, which will be used in our second approach to estimate the gradient sign vector.

Under some assumptions on the optimization objective and the hierarchical partitioning $\mathcal{T}$ of the search space, optimistic methods enjoy a finite-time bound on their *regret* R_t defined as

$$R_t = g(\boldsymbol{q}^*) - g(\boldsymbol{q}(t)) \;, \tag{9}$$

where $\boldsymbol{q}(t)$ is the best found solution by the optimistic method after t steps. The challenge is how to align the search space such that these assumptions hold. In the following, we show that these assumptions can be satisfied for our optimization objective (Eq. 8). In particular, when $g(\boldsymbol{q})$ is the directional derivative function $D_{\boldsymbol{q}}L(\boldsymbol{x}, y)$, and $\mathcal{H}$'s vertices are aligned on a 1-dimensional line according to the Gray code ordering, then we can construct an optimistic algorithm with a finite-time bound on its regret. To demonstrate this, we adopt the Simultaneous Optimistic Optimization framework by Munos (2011) and the assumptions therein.

For completeness, we reproduce (Munos, 2011)'s basic definitions and assumptions with respect to our notation. At the same time we show how the gradient sign estimation problem (Eq. 8) satisfies them based on the second property of the directional derivative as follows.

Definition 2 (Semi-metric)**.** *We assume that $\kappa : \mathcal{H} \times \mathcal{H} \to \mathbb{R}^+$ is such that for all $\boldsymbol{p}, \boldsymbol{q} \in \mathcal{H}$, we have $\kappa(\boldsymbol{p}, \boldsymbol{q}) = \kappa(\boldsymbol{q}, \boldsymbol{p})$ and $\kappa(\boldsymbol{p}, \boldsymbol{q}) = 0$ if and only if $\boldsymbol{p} = \boldsymbol{q}$.*

Definition 3 (Near-optimality dimension)**.** *The near-optimal dimension is the smallest $d \ge 0$ such that there exists $C > 0$ such that for any $\varepsilon > 0$, the maximal number of disjoint κ-balls of radius $\upsilon\varepsilon$ and center in $\mathcal{H}_\varepsilon$ is less than $C\varepsilon^{-d}$.*

Property 3 (Local smoothness of $D_{\boldsymbol{q}}L(\boldsymbol{x}, y)$)**.** *For any input/label pair $(\boldsymbol{x}, y)$, there exists at least a global optimizer $\boldsymbol{q}^* \in \mathcal{H}$ of $D_{\boldsymbol{q}}L(\boldsymbol{x}, y)$ (i.e., $D_{\boldsymbol{q}^*}L(\boldsymbol{x}, y) = \sup_{\boldsymbol{q} \in \mathcal{H}} D_{\boldsymbol{q}}L(\boldsymbol{x}, y)$) and for all $\boldsymbol{q} \in \mathcal{H}$,*

$$D_{\boldsymbol{q}^*}L(\boldsymbol{x}, y) - D_{\boldsymbol{q}}L(\boldsymbol{x}, y) \le \kappa(\boldsymbol{q}, \boldsymbol{q}^*) \;.$$

Refer to Figure 5 for a pictorial proof of Property 3.

Assumption 1 (Bounded diameters)**.** *There exists a decreasing a decreasing sequence $\omega(h) > 0$, such that for any depth $0 \le h < n$, for any cell $\mathcal{H}_{h,i}$ of depth h, we have $\sup_{\boldsymbol{q} \in \mathcal{H}_{h,i}} \kappa(\boldsymbol{q}_{h,i}, \boldsymbol{q}) \le \omega(h)$.*

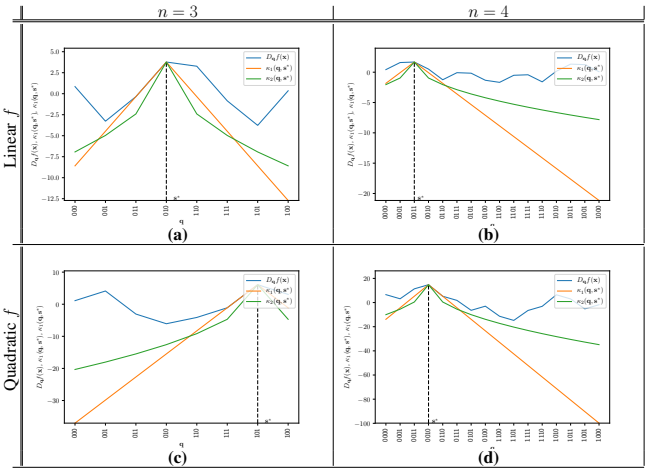


Figure 5. The directional derivative $D_{\boldsymbol{q}}f(\boldsymbol{x})$ of some function f at a point $\boldsymbol{x}$ in the direction of a binary vector $\boldsymbol{q} \in \mathcal{H} \equiv \{-1, +1\}^n$ is locally smooth around the gradient sign vector, $\boldsymbol{q}^* = \operatorname{sign}(\nabla_{\boldsymbol{x}} f(\boldsymbol{x})) \in \mathcal{H}$, when $\mathcal{H}$ is ordered over one coordinate as a sequence of Gray codes. The plots show the local smoothness property—with two semi-metrics $\kappa_1(\cdot, \cdot)$ and $\kappa_2(\cdot, \cdot)$—of the directional derivative of functions f of the form $\mathbf{c}^T\boldsymbol{x}$ and $\boldsymbol{x}^T Q \boldsymbol{x}$ for $n \in \{3, 4\}$, as tabulated in Plots (a), (c) and (b, d), respectively. The local smoothness is evident as $D_{\boldsymbol{q}^*}f(\boldsymbol{x}) - D_{\boldsymbol{q}}f(\boldsymbol{x}) \le \kappa_1(\boldsymbol{q}, \boldsymbol{q}^*)$ and $D_{\boldsymbol{q}^*}f(\boldsymbol{x}) - D_{\boldsymbol{q}}f(\boldsymbol{x}) \le \kappa_2(\boldsymbol{q}, \boldsymbol{q}^*)$ for all $\boldsymbol{q} \in \mathcal{H}$. The semi-metrics $\kappa_1(\boldsymbol{q}, \boldsymbol{q}^*)$ and $\kappa_2(\boldsymbol{q}, \boldsymbol{q}^*)$ have the form $K|\texttt{rank}_{Gray}(\boldsymbol{q}) - \texttt{rank}_{Gray}(\boldsymbol{q}^*)|^\alpha$, where $\texttt{rank}_{Gray}(\cdot)$ refers to the rank of an n-binary code in the Gray ordering of n-binary codes (e.g., $\texttt{rank}_{Gray}([-1, -1, +1, -1]) = 4$), $K > 0$, and $\alpha > 0$. With this property at hand, we employ the optimism in the face of uncertainty principle in GOO to maximize $D_{\boldsymbol{q}}f(\boldsymbol{x})$ over $\mathcal{H}$. For legibility, we replaced -1 with 0 when enumerating $\mathcal{H}$ on the x-axis.

To see how Assumption 1 is met, refer to Figure 7.

Assumption 2 (Well-shaped cells)**.** *There exists $\upsilon > 0$ such that for any depth $0 \le h < n$, any cell $\mathcal{H}_{h,i}$ contains a κ-ball of radius $\upsilon\omega(h)$ centered in $\boldsymbol{q}_{h,i}$.*

To see how Assumption 2 is met, refer to Figure 7. With the above assumptions satisfied, we propose the Gray-code Optimistic Optimization (GOO), which is an instantiation of (Munos, 2011, Algorithm 2) tailored to our optimization problem (Eq. 8) over a 1-dimensional alignment of $\mathcal{H}$ using the Gray code ordering. The pseudocode is outlined in Algorithm 1. The following theorem bounds GOO's regret.

Theorem 1. *Regret Convergence of GOO Let us write $h(t)$ the smallest integer h such that*

$$Ch_{max}(t) \sum_{l=0}^{h} \omega(l)^{-d} + 1 \ge t \;. \tag{10}$$

Then, with $g(\boldsymbol{q}) = D_{\boldsymbol{q}}L(\boldsymbol{x}, y)$, the regret of GOO (Algo-

Algorithm 1 Gray-code Optimistic Optimization (GOO)

Input:

$g : \mathcal{H} \to \mathbb{R}$	: the black-box linear function to be maximized over the binary hypercube $\mathcal{H} \equiv \{-1, +1\}^n$
$h_{max}(t)$	: the maximum depth function which limits the tree to grow up to $h_{max}(t)+1$ after t node expansions

Initialization: Set $t = 1$, $\mathcal{T}_t = \{(0, 0)\}$ (root node). Align $\mathcal{H}$ over $\mathcal{T}_t$ using the Gray code ordering.
while True **do**
  $v_{max} \leftarrow -\infty$
  for $h = 0$ **to** $\min(depth(\mathcal{T}_t), h_{max}(t))$ **do**
    Among all leaves $(h, j) \in \mathcal{L}_t$ of depth h, select

$$(h, i) \in \underset{(h,j)\in\mathcal{L}_t}{\arg\max}\ g(\boldsymbol{q}_{h,i})$$

    if $g(\boldsymbol{q}_{h,i}) \geq v_{max}$ **then**
      Expand this node: add to $\mathcal{T}_t$ the two children $(h + 1, i_k)_{1\leq k\leq 2}$
      $v_{max} \leftarrow g(\boldsymbol{q}_{h,i})$
      $t \leftarrow t + 1$
      if query budget is exhaused **then**
        return the best found solution $\boldsymbol{q}(t)$.
      end if
    end if
  end for
end while

rithm 1) is bounded as

$$R_t \leq \omega(\min(h(t), h_{max}(t) + 1))$$

Proof. We have showed that our objective function (Eq. 8) and the hierarchical partitioning of $\mathcal{H}$ following the Gray code ordering confirm to Property 3 and Assumptions 1 and 2. The $+1$ term in Eq. 10 is to accommodate the evaluation of node $(n, 0)$ before growing the space-partitioning tree $\mathcal{T}$—see Figure 7. The rest follows from the proof of (Munos, 2011, Theorem 2). □

Despite being theoretically-founded, GOO is slow in practice. This is expected since it is a global search technique that considers all the 2^n vertices of the n-dimensional hypercube $\mathcal{H}$. Recall that we are looking for adversarially helpful solution $\boldsymbol{q}$ that may not be necessarily optimal.

Empirical Evaluation of the Approaches on a Set of Toy Problems

We tested both GOO and SignHunter (along with Maurer's and Elimination)[2] on a set of toy problems and found that SignHunter performs significantly better than GOO, while Maurer's and Elimination were sensitive to the approximation error—see Figure 6. We remind the reader that Maurer's and Elimination are two search strategies for $\boldsymbol{q}^*$ using the Hamming oracle. After response $r^{(i)}$ to query $\boldsymbol{q}^{(i)}$, Elimination eliminates all binary codes $\boldsymbol{q} \in \mathcal{H}$ with $||\boldsymbol{q} - \boldsymbol{q}^{(i)}||_H \neq r^{(i)}$ in an iterative manner. Note that Elimination is a naive technique that is not scalable with n. For Maurer's, we refer the reader to (Maurer, 2009).

[2] See Appendix C for details on these algorithms.

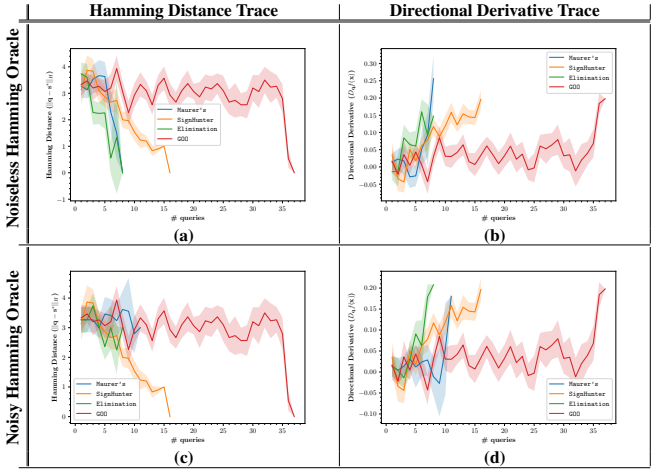
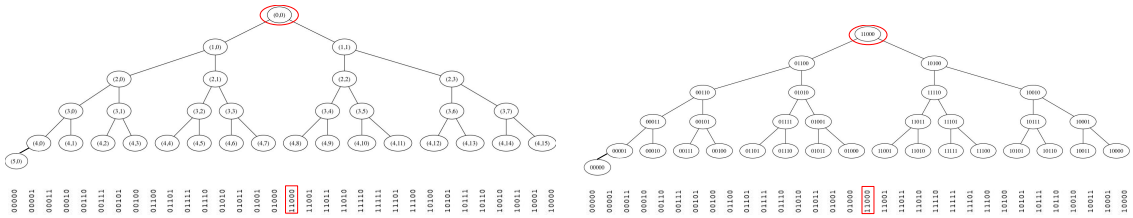


Figure 6. Noiseless vs. Noisy Hamming Oracle: The trace of the Hamming distance (first column, where the lower the trace the better) and directional derivative (second column, where the higher the trace the better) values of `Elimination` and `Maurer's` queries, when given access to a noiseless/ideal (first row) and noisy Hamming oracles (second row)—through a directional derivative approximation as discussed in Approach 2—for a synthetic function f of the form $\boldsymbol{x}^T Q \boldsymbol{x}$ with $n = 7$. We expect the traces to go up and down as they explore the 2^n search space. The end of an algorithm's trace represents the value of the Hamming distance (directional derivative) for the first column (for the second column) at the algorithm's solution. For comparison, we also plot `GOO` and `SignHunter`'s traces. Note that the performance of `GOO` and `SignHunter` is the same in both noiseless and noisy cases as both algorithms operate directly on the directional derivative approximation rather than the noiseless/noisy Hamming oracle. In the case of noiseless Hamming oracle, both `Elimination` and `Maurer's` finds the optimal vector $\boldsymbol{q}^* \in \mathcal{H} \equiv \{-1, +1\}^7$ with # queries ≤ 7—their traces end at most at 8 just to show that the algorithm's solution achieves a zero Hamming distance as shown in Plot (a), which corresponds to the maximum directional derivative in Plot (b). With a noisy Hamming oracle, these algorithm break as shown in Plots (c) and (d): taking more than n queries and returning sub-optimal solutions—e.g., `Maurer's` returns on average a three-Hamming-distance solution. On the other hand, `GOO` and `SignHunter` achieve a zero Hamming distance in both cases at the expense of being less query efficient. While being theoretically-founded, `GOO` is slow as it employs a global search over the 2^n space. Despite `SignHunter`'s local search, it converges to the optimal solution after 2×7 queries in accordance with Theorem 3. The solid curves indicate the corresponding averaged trace surrounded by a 95%-confidence bounds using 30 independent runs. For convenience, we plot the symmetric bounds where in fact they should be asymmetric in Plots (a) and (c) as the Hamming distance's range is $\mathbb{Z}_0^+$.



(a) (h, i) view (b) $\boldsymbol{q}_{h,i}$ view

Figure 7. Illustration of the proposed Gray-ordering based partitioning (fully expanded) tree $\mathcal{T}$ of the search space $\mathcal{H} = \{-1, +1\}^n$—with $n = 5$—used in the `G`ray-code `O`ptimistic `O`ptimization (`GOO`). The plots are two different views of the same tree. Plot (a) displays the node name (h, i), while Plot (b) shows its representative binary code $\boldsymbol{q}_{h,i} \in \mathcal{H}$. For brevity, we replaced -1s with 0s. The red oval and rectangle highlights the tree's root and its corresponding binary code, respectively. Consider the node $(2, 1)$ whose representative code is $\boldsymbol{q}_{2,1} = [-1, +1, -1, +1, -1]$ and its corresponding subspace $\mathcal{H}_{2,1} = \{\boldsymbol{q}_{2,1}, \boldsymbol{q}_{3,2}, \boldsymbol{q}_{3,3}, \boldsymbol{q}_{4,4}, \boldsymbol{q}_{4,5}, \boldsymbol{q}_{4,6}, \boldsymbol{q}_{4,7}\}$. The same reasoning applies to the rest of the nodes. To maintain a valid binary partition tree, one can ignore the anomaly leaf node $(5, 0)$, this corresponds to the code $\boldsymbol{q}_{5,0} = [-1, -1, -1, -1, -1]$, which in practice can be evaluated prior to building the tree. Let us consider the nodes at depth $h = 2$, observe that for all $i \in \{0, 1, 2, 3\}$ 1) $|\mathcal{H}_{2,i}| = 7$; 2) $\boldsymbol{q}_{2,i}$ is centered around the other 6 members of $\mathcal{H}_{2,i}$ in the Gray code ordering; and that 3) $\mathcal{H}_{2,i}$ constitutes a contiguous block of codes along the 1-dimensional alignment shown below the tree. Thus, it suffices to define a semi-metric based on the corresponding indices of the codes along this alignment. For a given depth h, the index of any code $\boldsymbol{q} \in \mathcal{H}_{h,i}$ is at most $\omega(h) = \sup_j |\mathcal{H}_{h+1,j}|$, which establishes Assumption 2. Assumption 3 follows naturally from the fact that nodes at a given depth h partition the search space $\mathcal{H}$ equally (e.g., $|\mathcal{H}_{2,i}| = 7$ for all i).

Appendix C. Proofs for Theorems in the Main Paper

Along with the proofs, we restate the theorems for completeness.

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. The key element of this proof is that the Hamming distance between two n-dimensional binary codes $\boldsymbol{q}, \boldsymbol{q}^* \in \mathcal{H}$ can be written as

$$r = ||\boldsymbol{q} - \boldsymbol{q}^*||_H = \frac{1}{2}(n - \boldsymbol{q}^T \boldsymbol{q}^*) \; . \tag{11}$$

Let Q be an $n \times n$ matrix where the ith row is the ith query code $\boldsymbol{q}^{(i)}$. Likewise, let $r^{(i)}$ be the corresponding ith query response, and $\boldsymbol{r}$ is the concatenating vector. In matrix form, we have

$$\boldsymbol{q}^* = Q^{-1}(n\mathbf{1}_n - 2\boldsymbol{r}) \; ,$$

where Q is invertible if we construct linearly independent queries $\{\boldsymbol{q}^{(i)}\}_{1 \leq i \leq n}$. □

In Figure 8, we plot the bounds above for $n = \{1, \ldots, 10\}$, along with two search strategies for $\boldsymbol{q}^*$ using the Hamming oracle: i) `Maurer's` (Maurer, 2009); and ii) search by `Elimination` which, after response $r^{(i)}$ to query $\boldsymbol{q}^{(i)}$, eliminates all binary codes $\boldsymbol{q} \in \mathcal{H}$ with $||\boldsymbol{q} - \boldsymbol{q}^{(i)}||_H \neq r^{(i)}$ in an iterative manner. Note that `Elimination` is a naive technique that is not scalable with n.

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. 2 in the main paper), `SignHunter` is at least as effective as `FGSM` (Goodfellow et al., 2015) in crafting adversarial examples.*

Proof. Recall that the ith coordinate of the gradient sign vector can be recovered as outlined in Eq. 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. 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. 2 in the main paper) is good enough (or at least a rank-preserving). □

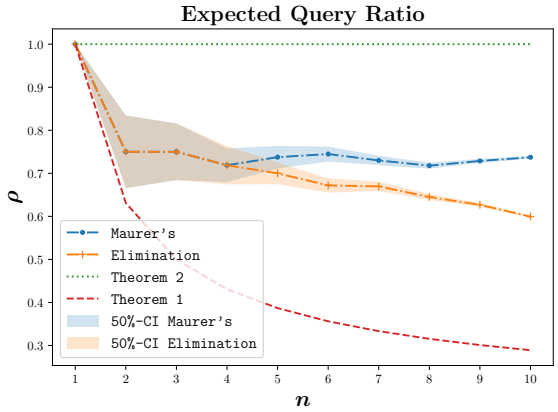
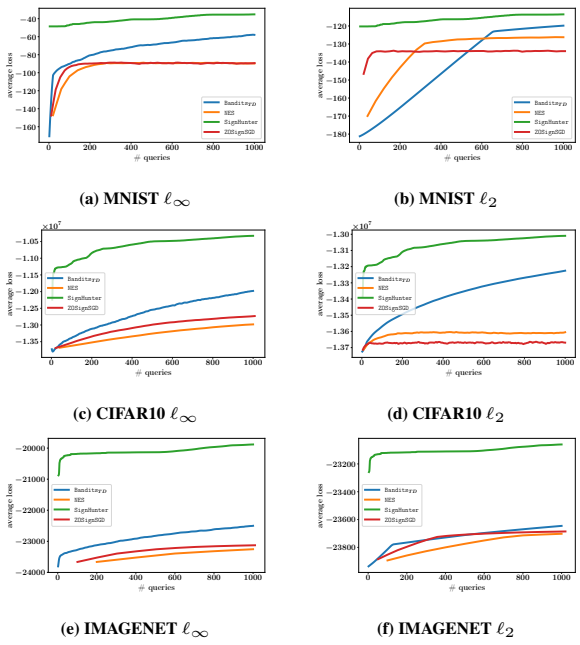


Figure 8. Expected Query Ratios for $n = \{1, \ldots, 10\}$ with the *noiseless* Hamming oracle $\mathcal{O}$.

Appendix D. Experiments Setup

This section outlines the experiments setup as follows. Figure 9 shows the performance of the considered algorithms on a synthetic concave loss function after tuning their hyperparameters. 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 $\frac{\boldsymbol{x}_{min}+\boldsymbol{x}_{max}}{2}$. Thus, `SignHunter` will recover the true gradient sign with as few queries as possible (recall the example in Section 4.1 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. Tables 2, 3, 4, and 5 outline the algorithms' hyperparameters, while Table 1 describes the general setup for the experiments.



(a) MNIST ℓ_∞ **(b) MNIST ℓ_2**

(c) CIFAR10 ℓ_∞ **(d) CIFAR10 ℓ_2**

(e) IMAGENET ℓ_∞ **(f) IMAGENET ℓ_2**

Figure 9. Tuning testbed for the attacks. A synthetic loss function was used to tune the performance of the attacks over a random sample of 25 images for each dataset and ℓ_p perturbation constraint. The plots above show the average performance of the tuned attacks on the synthetic loss function $L(\boldsymbol{x}, y) = -(\boldsymbol{x} - \boldsymbol{x}^*)^T(\boldsymbol{x} - \boldsymbol{x}^*)$, where $\boldsymbol{x}^* = \frac{\boldsymbol{x}_{min}+\boldsymbol{x}_{max}}{2}$ using a query limit of 1000 queries for each image. Note that in all, `Bandits`$_{TD}$ 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.

Table 1. General setup for all the attacks

	Value					
	MNIST		CIFAR10		IMAGENET	
	ℓ_∞	ℓ_2	ℓ_∞	ℓ_2	ℓ_∞	ℓ_2
Parameter						
ϵ (allowed perturbation)	0.3	3	12	127	0.05	5
Max allowed queries	10000					
Evaluation/Test set size	1000					
Data (pixel value) Range	[0,1]		[0,255]		[0,1]	

Table 2. Hyperparameters setup for `NES`

	Value					
	MNIST		CIFAR10		IMAGENET	
	ℓ_∞	ℓ_2	ℓ_∞	ℓ_2	ℓ_∞	ℓ_2
Hyperparameter						
δ (finite difference probe)	0.1	0.1	2.55	2.55	0.1	0.1
η (image ℓ_p learning rate)	0.1	1	2	127	0.02	2
q (number of finite difference estimations per step)	10	20	20	4	100	50

Table 3. Hyperparameters setup for `ZO-SignSGD`

	Value					
	MNIST		CIFAR10		IMAGENET	
	ℓ_∞	ℓ_2	ℓ_∞	ℓ_2	ℓ_∞	ℓ_2
Hyperparameter						
δ (finite difference probe)	0.1	0.1	2.55	2.55	0.1	0.1
η (image ℓ_p learning rate)	0.1	0.1	2	2	0.02	0.004
q (number of finite difference estimations per step)	10	20	20	4	100	50

Table 4. Hyperparameters setup for `Bandits`$_{TD}$

	Value					
	MNIST		CIFAR10		IMAGENET	
	ℓ_∞	ℓ_2	ℓ_∞	ℓ_2	ℓ_∞	ℓ_2
Hyperparameter						
η (image ℓ_p learning rate)	0.03	0.01	5	12	0.01	0.1
δ (finite difference probe)	0.1	0.1	2.55	2.55	0.1	0.1
τ (online convex optimization learning rate)	0.001	0.0001	0.0001	1e-05	0.0001	0.1
Tile size (data-dependent prior)	8	10	20	20	50	50
ζ (bandit exploration)	0.01	0.1	0.1	0.1	0.01	0.1

Table 5. Hyperparameters setup for `SignHunter`

	Value					
	MNIST		CIFAR10		IMAGENET	
	ℓ_∞	ℓ_2	ℓ_∞	ℓ_2	ℓ_∞	ℓ_2
Hyperparameter						
δ (finite difference probe)	0.3	3	12	127	0.05	5

Appendix E. Results of Adversarial Black-Box Examples Generation

This section shows results of our experiments in crafting adversarial black-box examples in the form of tables and performance traces, namely Figures 10, 11, and 12; and Tables 6, 7, and 8.

Table 6. Summary of attacks effectiveness on MNIST under ℓ_∞ and ℓ_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.

	Failure Rate		**Avg. # Queries**	
	ℓ_∞	ℓ_2	ℓ_∞	ℓ_2
Attack				
Bandits$_{TD}$	0.68	0.59	328.00	673.16
NES	0.63	0.63	235.07	**361.42**
SignHunter	**0.00**	**0.04**	**11.06**	1064.22
ZOSignSGD	0.63	0.75	157.00	881.08

Table 7. Summary of attacks effectiveness on CIFAR10 under ℓ_∞ and ℓ_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.

	Failure Rate		**Avg. # Queries**	
	ℓ_∞	ℓ_2	ℓ_∞	ℓ_2
Attack				
Bandits$_{TD}$	0.95	0.39	432.24	1201.85
NES	0.37	0.67	312.57	**496.99**
SignHunter	**0.07**	**0.21**	**121.00**	692.39
ZOSignSGD	0.37	0.80	161.28	528.35

Table 8. Summary of attacks effectiveness on IMAGENET under ℓ_∞ and ℓ_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.

	Failure Rate		**Avg. # Queries**	
	ℓ_∞	ℓ_2	ℓ_∞	ℓ_2
Attack				
Bandits$_{TD}$	0.07	**0.11**	1010.05	1635.55
NES	0.26	0.42	1536.19	1393.86
SignHunter	**0.02**	0.23	**578.56**	1985.55
ZOSignSGD	0.23	0.52	1054.98	**931.15**

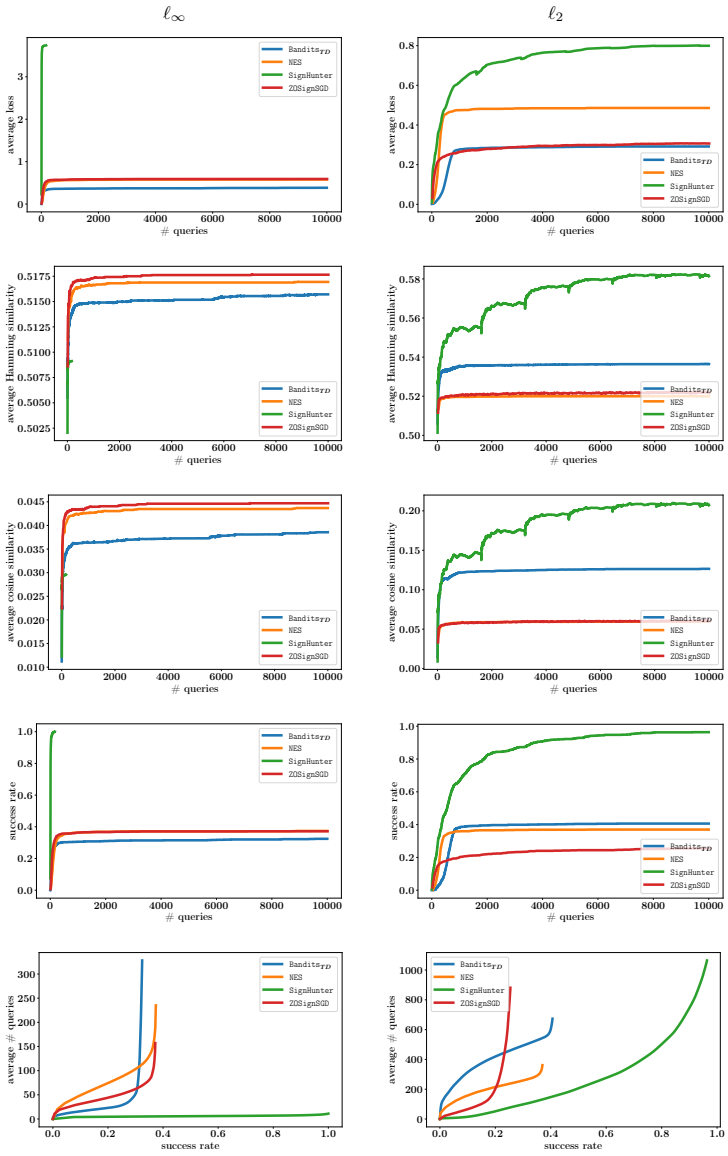


Figure 10. Performance curves of attacks on MNIST for ℓ_∞ (first column) and ℓ_2 (second column) perturbation constraints. Plots of *Avg. Loss* row reports the loss as a function of the number of queries averaged over all images. The *Avg. Hamming Similarity* row shows the Hamming similarity of the sign of the attack's estimated gradient $\hat{\boldsymbol{g}}$ with true gradient's sign $\boldsymbol{q}^*$, computed as $1 - ||\text{sign}(\hat{\boldsymbol{g}}) - \boldsymbol{q}^*||_H/n$ and averaged over all images. Likewise, plots of the *Avg. Cosine Similarity* row show the normalized dot product of $\hat{\boldsymbol{g}}$ and $\boldsymbol{g}^*$ 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.

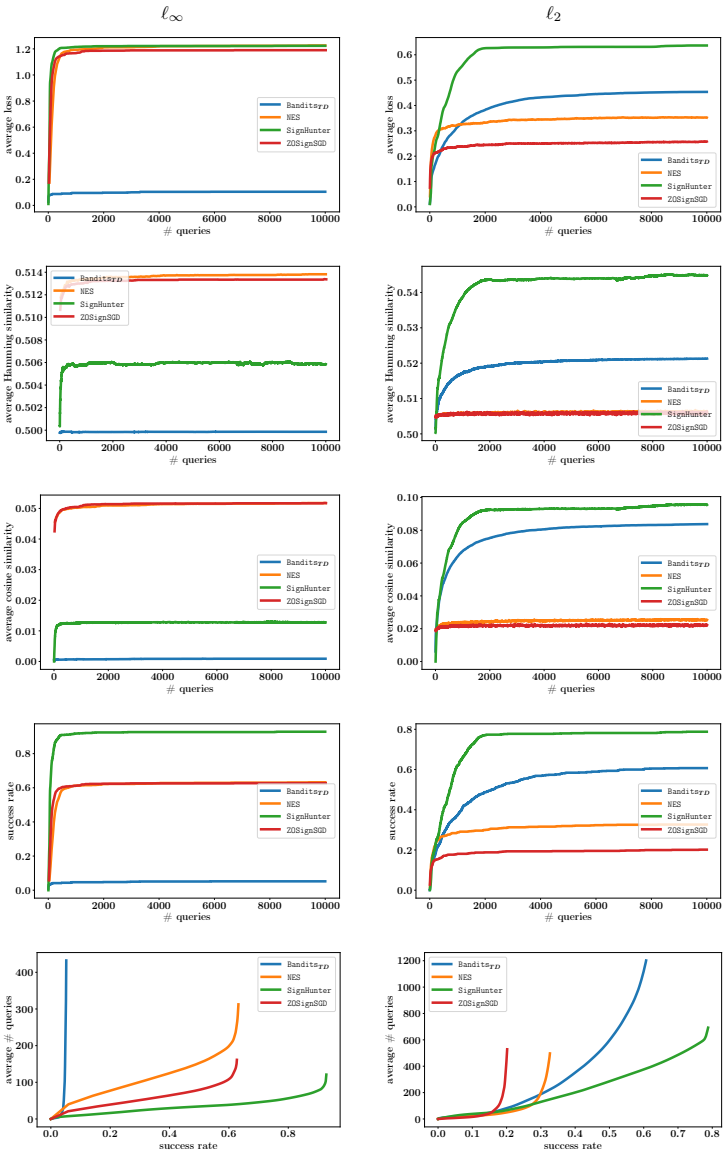


Figure 11. Performance curves of attacks on CIFAR10 for ℓ_∞ (first column) and ℓ_2 (second column) perturbation constraints. Plots of *Avg. Loss* row reports the loss as a function of the number of queries averaged over all images. The *Avg. Hamming Similarity* row shows the Hamming similarity of the sign of the attack's estimated gradient $\hat{\boldsymbol{g}}$ with true gradient's sign $\boldsymbol{q}^*$, computed as $1 - ||\text{sign}(\hat{\boldsymbol{g}}) - \boldsymbol{q}^*||_H/n$ and averaged over all images. Likewise, plots of the *Avg. Cosine Similarity* row show the normalized dot product of $\hat{\boldsymbol{g}}$ and $\boldsymbol{g}^*$ 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.

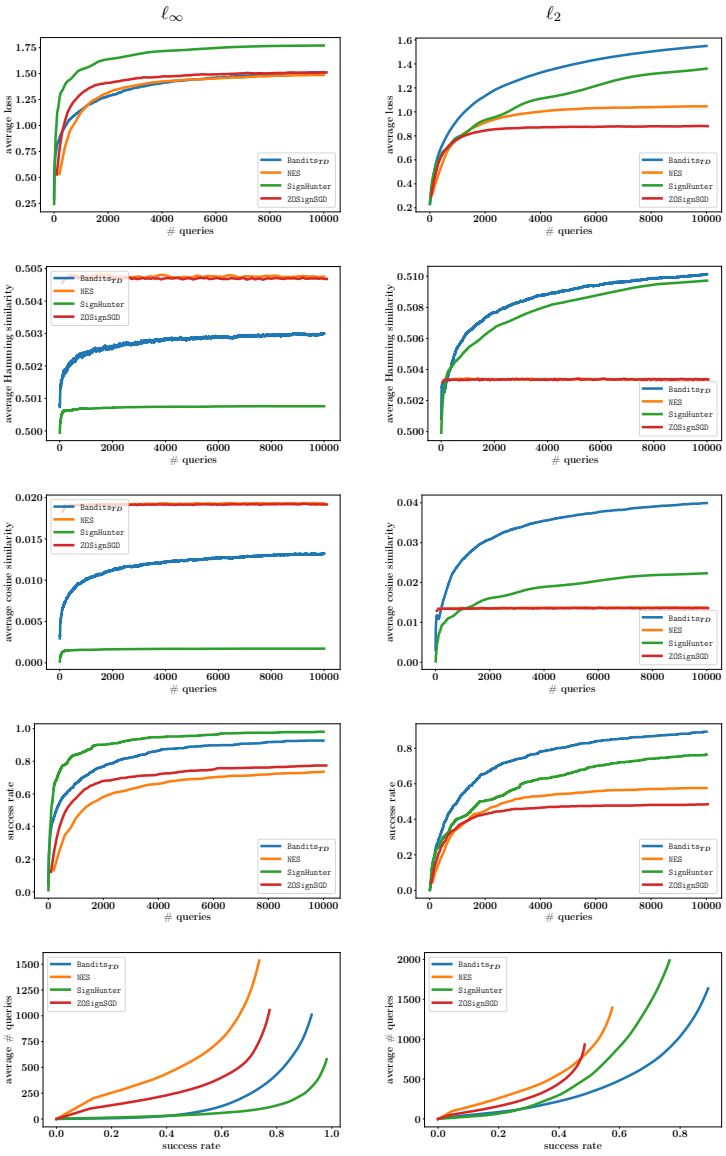


Figure 12. Performance curves of attacks on IMAGENET for ℓ_∞ (first column) and ℓ_2 (second column) perturbation constraints. Plots of *Avg. Loss* row reports the loss as a function of the number of queries averaged over all images. The *Avg. Hamming Similarity* row shows the Hamming similarity of the sign of the attack's estimated gradient $\hat{\boldsymbol{g}}$ with true gradient's sign $\boldsymbol{q}^*$, computed as $1 - ||\text{sign}(\hat{\boldsymbol{g}}) - \boldsymbol{q}^*||_H / n$ and averaged over all images. Likewise, plots of the *Avg. Cosine Similarity* row show the normalized dot product of $\hat{\boldsymbol{g}}$ and $\boldsymbol{g}^*$ 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 F. Public Black-Box Challenge Results

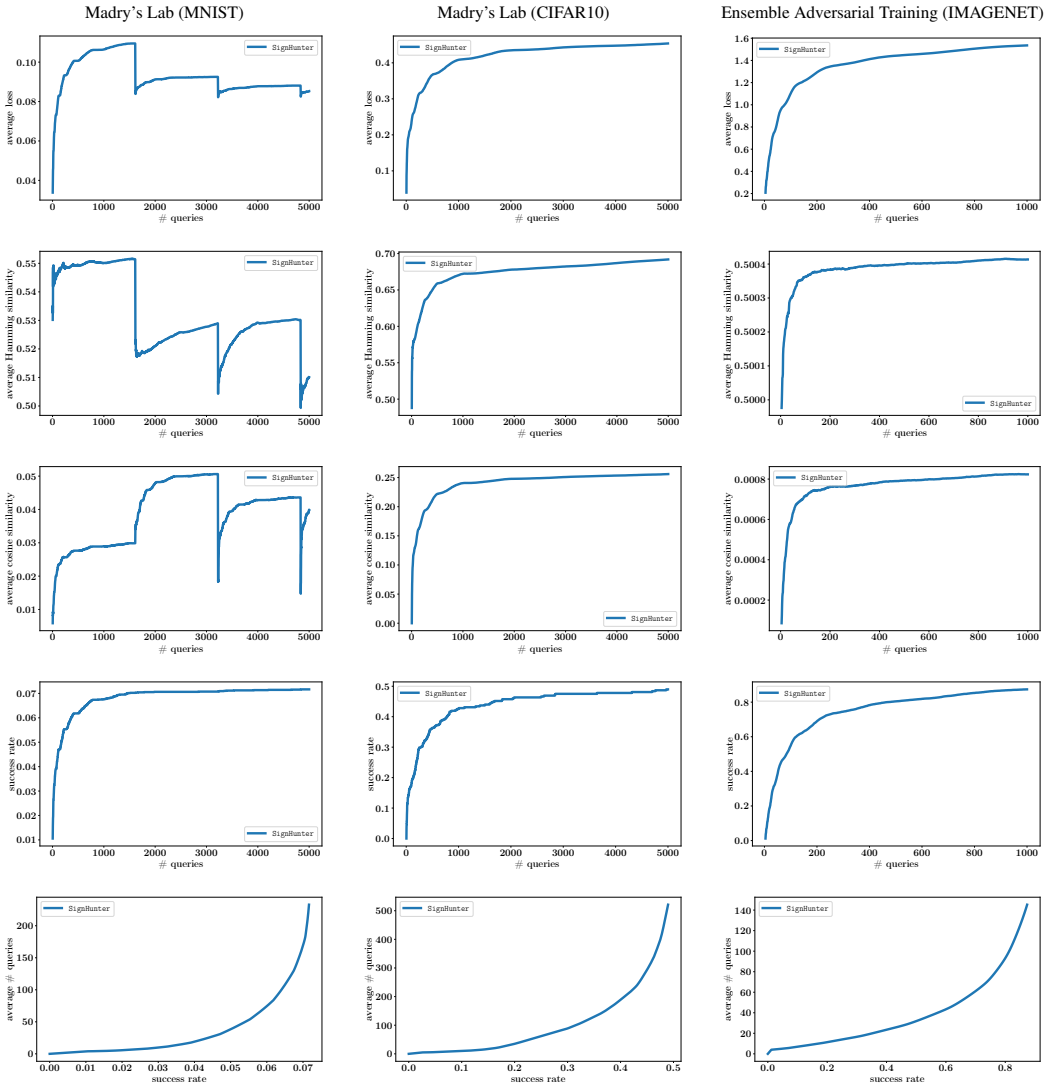


Figure 13. Performance curves of attacks on the public black-box challenges for MNIST (first column), CIFAR10 (second column) and IMAGENET (third column). Plots of *Avg. Loss* row reports the loss as a function of the number of queries averaged over all images. The *Avg. Hamming Similarity* row shows the Hamming similarity of the sign of the attack's estimated gradient $\hat{\boldsymbol{g}}$ with true gradient's sign $\boldsymbol{q}^*$, computed as $1 - ||\text{sign}(\hat{\boldsymbol{g}}) - \boldsymbol{q}^*||_H/n$ and averaged over all images. Likewise, plots of the *Avg. Cosine Similarity* row show the normalized dot product of $\hat{\boldsymbol{g}}$ and $\boldsymbol{g}^*$ 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 $5,000$ 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 G. Estimating Hamming Oracle

This section illustrates our experiment on the distribution of the magnitudes of gradient coordinates as summarized in Figure 14. *How to read the plots:* Consider the first histogram in Plot (a) from below; it corresponds to the 1000^{th} image from the sampled MNIST evaluation set, plotting the histogram of the values $\{|\partial L(\boldsymbol{x}, y)/\partial x_i|\}_{1 \leq i \leq n}$, 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 ~ 0.055. Thus, a Monte Carlo estimate of the mean of $\{|\partial L(\boldsymbol{x}, y)/\partial x_i|\}_{1 \leq i \leq n}$ would be an appropriate approximation. We release these figures in the form of TensorBoard logs.

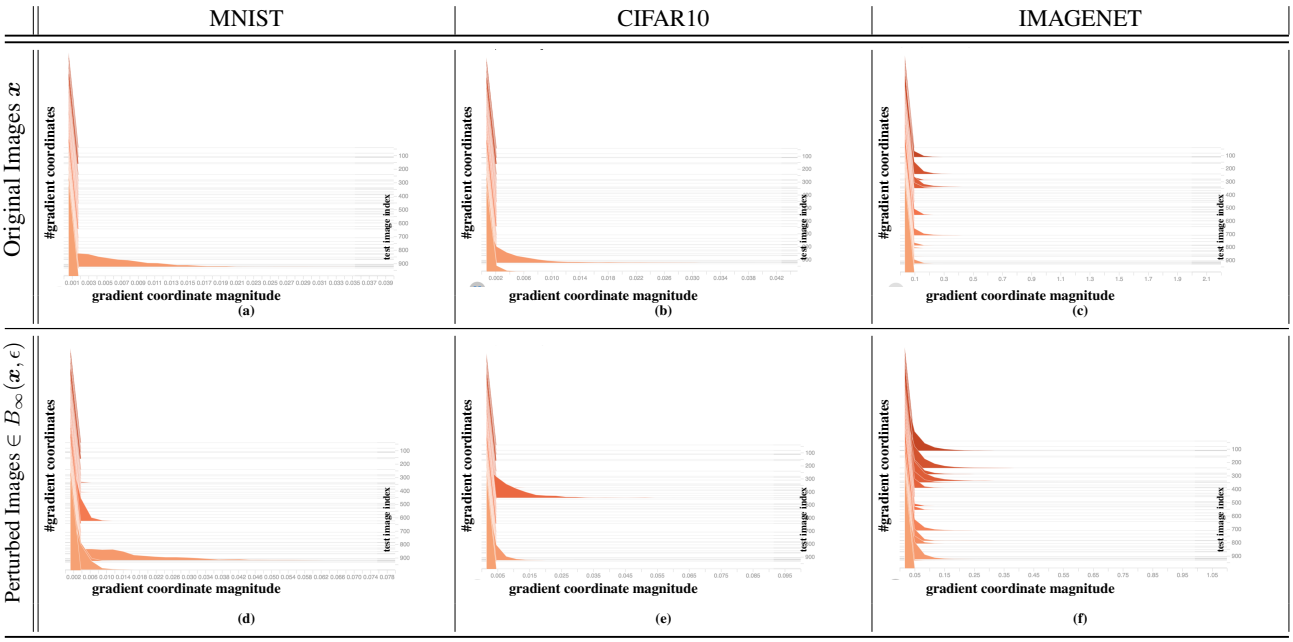


Figure 14. Magnitudes of gradient coordinates are concentrated: Plots (a), (b), and (c) show histograms of the magnitudes of gradient coordinates of the loss function $L(\boldsymbol{x}, y)$ with respect to the input point (image) $\boldsymbol{x}$ for MNIST, CIFAR10, and IMAGENET neural net models over 1000 images from the corresponding evaluation set, respectively. Plots (d), (e), (f) show the same but at input points (images) sampled randomly within $B_\infty(\boldsymbol{x}, \epsilon)$: the ℓ_∞-ball of radius $\epsilon = 0.3$, 12, and 0.05 around the images in Plots (a), (b), and (c), respectively.

References

Al-Dujaili, A. and Suresh, S. Embedded bandits for large-scale black-box optimization. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

Al-Dujaili, A. and Suresh, S. Multi-objective simultaneous optimistic optimization. *Information Sciences*, 424:159–174, 2018.

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.

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.

Kocsis, L. and Szepesvári, C. Bandit based monte-carlo planning. In *European conference on machine learning*, pp. 282–293. Springer, 2006.

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.

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

Munos, R. Optimistic optimization of a deterministic function without the knowledge of its smoothness. In *Advances in neural information processing systems*, pp. 783–791, 2011.