Square Attack: a query-efficient black-box adversarial attack via random search

Maksym Andriushchenko*[1], Francesco Croce*[2],
Nicolas Flammarion[1], and Matthias Hein[2]

[1] EPFL
[2] University of Tübingen

Abstract. We propose the *Square Attack*, a score-based black-box l_2- and l_∞-adversarial attack that does not rely on local gradient information and thus is not affected by gradient masking. Square Attack is based on a randomized search scheme which selects localized square-shaped updates at random positions so that at each iteration the perturbation is situated approximately at the boundary of the feasible set. Our method is significantly more query efficient and achieves a higher success rate compared to the state-of-the-art methods, especially in the untargeted setting. In particular, on ImageNet we improve the average query efficiency in the untargeted setting for various deep networks by a factor of at least 1.8 and up to 3 compared to the recent state-of-the-art l_∞-attack of Al-Dujaili & O'Reilly (2020). Moreover, although our attack is *black-box*, it can also outperform gradient-based *white-box* attacks on the standard benchmarks achieving a new state-of-the-art in terms of the success rate. The code of our attack is available at `https://github.com/max-andr/square-attack`.

1 Introduction

Adversarial examples are of particular concern when it comes to applications of machine learning which are safety-critical. Many defenses against adversarial examples have been proposed [26,62,45,5,37,1,7] but with limited success, as new more powerful attacks could break many of them [12,4,40,14,63]. In particular, gradient obfuscation or masking [4,40] is often the reason why seemingly robust models turn out to be non-robust in the end. Gradient-based attacks are most often affected by this phenomenon (white-box attacks but also black-box attacks based on finite difference approximations [40]). Thus it is important to have attacks which are based on different principles. Black-box attacks have recently become more popular [41,9,51] as their attack strategies are quite different from the ones employed for adversarial training, where often PGD-type attacks [37] are used. However, a big weakness currently is that these black-box attacks need to query the classifier too many times before they find adversarial examples, and their success rate is sometimes significantly lower than that of white-box attacks.

*Equal contribution.

In this paper we propose Square Attack, a score-based adversarial attack, i.e. it can query the probability distribution over the classes predicted by a classifier but has no access to the underlying model. The Square Attack exploits random search[3] [46,48] which is one of the simplest approaches for black-box optimization. Due to a particular sampling distribution, it requires significantly fewer queries compared to the state-of-the-art black-box methods (see Fig. 1) in the score-based threat model while outperforming them in terms of *success rate*, i.e. the percentage of successful adversarial examples. This is achieved by a combination of a particular initialization strategy and our square-shaped updates. We motivate why these updates are particularly suited to attack neural networks and provide convergence guarantees for a variant of our method. In an extensive evaluation with untargeted and targeted attacks, three datasets (MNIST, CIFAR-10, ImageNet), normal and robust models, we show that Square Attack outperforms state-of-the-art methods in the l_2- and l_∞-threat model.

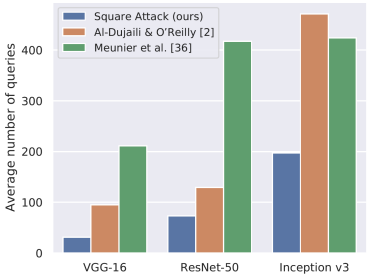

Fig. 1. Avg. number of queries of successful untargeted l_∞-attacks on three ImageNet models for three score-based black-box attacks. Square Attack outperforms all other attacks by large margin

2 Related Work

We discuss black-box attacks with l_2- and l_∞-perturbations since our attack focuses on this setting. Although attacks for other norms, e.g. l_0, exist [41,19], they are often algorithmically different due to the geometry of the perturbations.

l_2- and l_∞-score-based attacks. Score-based black-box attacks have only access to the score predicted by a classifier for each class for a given input. Most of such attacks in the literature are based on gradient estimation through finite differences. The first papers in this direction [6,30,56] propose attacks which approximate the gradient by sampling from some noise distribution around the point. While this approach can be successful, it requires many queries of the classifier, particularly in high-dimensional input spaces as in image classification. Thus, improved techniques reduce the dimension of the search space via using the principal components of the data [6], searching for perturbations in the latent space of an auto-encoder [55] or using a low-dimensional noise distribution [31]. Other attacks exploit evolutionary strategies or random search, e.g. [3] use a genetic algorithm to generate adversarial examples and alleviate gradient masking as they can reduce the robust accuracy on randomization- and discretization-based defenses. The l_2-attack of [28] can be seen as a variant of random search which chooses the search directions in an orthonormal basis and tests up to two candidate updates at each step. However, their algorithm can have suboptimal query efficiency since it adds at every step only small (in l_2-norm)

[3] It is an iterative procedure different from random sampling inside the feasible region.

modifications, and suboptimal updates cannot be undone as they are orthogonal to each other. A recent line of work has pursued black-box attacks which are based on the observation that successful adversarial perturbations are attained at corners of the l_∞-ball intersected with the image space $[0, 1]^d$ [49,2,39]. Searching over the corners allows to apply discrete optimization techniques to generate adversarial attacks, significantly improving the query efficiency. Both [49] and [2] divide the image according to some coarse grid, perform local search in this lower dimensional space allowing componentwise changes only of $-\epsilon$ and ϵ, then refine the grid and repeat the scheme. In [2] such a procedure is motivated as an estimation of the gradient signs. Recently, [39] proposed several attacks based on evolutionary algorithms, using discrete and continuous optimization, achieving nearly state-of-the-art query efficiency for the l_∞-norm. In order to reduce the dimensionality of the search space, they use the "tiling trick" of [31] where they divide the perturbation into a set of squares and modify the values in these squares with evolutionary algorithms. A related idea also appeared earlier in [25] where they introduced black rectangle-shaped perturbations for generating adversarial occlusions. In [39], as in [31], both size and position of the squares are fixed at the beginning and not optimized. Despite their effectiveness for the l_∞-norm, these discrete optimization based attacks are not straightforward to adapt to the l_2-norm. Finally, approaches based on Bayesian optimization exist, e.g. [50], but show competitive performance only in a low-query regime.

Different threat and knowledge models. We focus on *l_p-norm-bounded* adversarial perturbations (for other perturbations such as rotations, translations, occlusions in the black-box setting see, e.g., [25]). Perturbations with *minimal* l_p-norm are considered in [15,55] but require significantly more queries than norm-bounded ones. Thus we do not compare to them, except for [28] which has competitive query efficiency while aiming at small perturbations.

In other cases the attacker has a different knowledge of the classifier. A more restrictive scenario, considered by *decision-based* attacks [9,16,27,11,13], is when the attacker can query only the decision of the classifier, but not the predicted scores. Other works use more permissive threat models, e.g., when the attacker already has a substitute model similar to the target one [44,57,17,23,52] and thus can generate adversarial examples for the substitute model and then transfer them to the target model. Related to this, [57] suggest to refine this approach by running a black-box gradient estimation attack in a subspace spanned by the gradients of substitute models. However, the gain in query efficiency given by such extra knowledge does not account for the computational cost required to train the substitute models, particularly high on ImageNet-scale. Finally, [35] use extra information on the target data distribution to train a model that predicts adversarial images that are then refined by gradient estimation attacks.

3 Square Attack

In the following we recall the definitions of the adversarial examples in the threat model we consider and present our black-box attacks for the l_∞- and l_2-norms.

Algorithm 1: The Square Attack via random search

Input: classifier f, point $x \in \mathbb{R}^d$, image size w, number of color channels c, l_p-radius ϵ, label $y \in \{1, \dots, K\}$, number of iterations N
Output: approximate minimizer $\hat{x} \in \mathbb{R}^d$ of the problem stated in Eq. (1)

$\hat{x} \leftarrow init(x)$, $l^* \leftarrow L(f(x), y)$, $i \leftarrow 1$
while $i < N$ ***and*** $\hat{x}$ *is not adversarial* **do**
  $h^{(i)} \leftarrow$ side length of the square to modify (according to some schedule)
  $\delta \sim P(\epsilon, h^{(i)}, w, c, \hat{x}, x)$ (see Alg. 2 and 3 for the sampling distributions)
  $\hat{x}_{\text{new}} \leftarrow$ Project $\hat{x} + \delta$ onto $\{z \in \mathbb{R}^d : \|z - x\|_p \leq \epsilon\} \cap [0, 1]^d$
  $l_{\text{new}} \leftarrow L(f(\hat{x}_{\text{new}}), y)$
  if $l_{new} < l^*$ **then** $\hat{x} \leftarrow \hat{x}_{\text{new}}$, $l^* \leftarrow l_{\text{new}}$;
  $i \leftarrow i + 1$
end

3.1 Adversarial Examples in the l_p-threat Model

Let $f : [0, 1]^d \to \mathbb{R}^K$ be a classifier, where d is the input dimension, K the number of classes and $f_k(x)$ is the predicted score that x belongs to class k. The classifier assigns class $\arg\max_{k=1,\dots,K} f_k(x)$ to the input x. The goal of an *untargeted* attack is to change the correctly predicted class y for the point x. A point $\hat{x}$ is called an *adversarial example* with an l_p-norm bound of ϵ for x if

$$\arg\max_{k=1,\dots,K} f_k(\hat{x}) \neq y, \quad \|\hat{x} - x\|_p \leq \epsilon \quad \text{and} \quad \hat{x} \in [0, 1]^d,$$

where we have added the additional constraint that $\hat{x}$ is an image. The task of finding $\hat{x}$ can be rephrased as solving the constrained optimization problem

$$\min_{\hat{x} \in [0,1]^d} L(f(\hat{x}), y), \quad \text{s.t.} \quad \|\hat{x} - x\|_p \leq \epsilon, \tag{1}$$

for a loss L. In our experiments, we use $L(f(\hat{x}), y) = f_y(\hat{x}) - \max_{k \neq y} f_k(\hat{x})$.

The goal of *targeted* attacks is instead to change the decision of the classifier to a particular class t, i.e., to find $\hat{x}$ so that $\arg\max_k f_k(\hat{x}) = t$ under the same constraints on $\hat{x}$. We further discuss the targeted attacks in Sup. E.1.

3.2 General Algorithmic Scheme of the Square Attack

Square Attack is based on *random search* which is a well known iterative technique in optimization introduced by Rastrigin in 1963 [46]. The main idea of the algorithm is to sample a random update δ at each iteration, and to add this update to the current iterate $\hat{x}$ if it improves the objective function. Despite its simplicity, random search performs well in many situations [60] and does not depend on gradient information from the objective function g.

Many variants of random search have been introduced [38,48,47], which differ mainly in how the random perturbation is chosen at each iteration (the original

scheme samples uniformly on a hypersphere of fixed radius). For our goal of crafting adversarial examples we come up with two sampling distributions specific to the l_∞- and the l_2-attack (Sec. 3.3 and Sec. 3.4), which we integrate in the classic random search procedure. These sampling distributions are motivated by both how images are processed by neural networks with convolutional filters and the shape of the l_p-balls for different p. Additionally, since the considered objective is non-convex when using neural networks, a good initialization is particularly important. We then introduce a specific one for better query efficiency.

Our proposed scheme differs from classical random search by the fact that the perturbations $\hat{x} - x$ are constructed such that for every iteration they lie on the boundary of the l_∞- or l_2-ball before projection onto the image domain $[0, 1]^d$. Thus we are using the perturbation budget almost maximally at each step. Moreover, the changes are localized in the image in the sense that at each step we modify just a small fraction of contiguous pixels shaped into **squares**. Our overall scheme is presented in Algorithm 1. First, the algorithm picks the side length $h^{(i)}$ of the square to be modified (step 3), which is decreasing according to an a priori fixed schedule. This is in analogy to the step-size reduction in gradient-based optimization. Then in step 4 we sample a new update δ and add it to the current iterate (step 5). If the resulting loss (obtained in step 6) is smaller than the best loss so far, the change is accepted otherwise discarded. Since we are interested in query efficiency, the algorithm stops as soon as an adversarial example is found. The time complexity of the algorithm is dominated by the evaluation of $f(\hat{x}_{\text{new}})$, which is performed at most N times, with N total number of iterations. We plot the resulting adversarial perturbations in Fig. 3 and additionally in Sup. E where we also show imperceptible perturbations.

We note that previous works [31,49,39] generate perturbations containing squares. However, while those use a fixed grid on which the squares are constrained, we optimize the position of the squares as well as the color, making our attack more flexible and effective. Moreover, unlike previous works, we motivate squared perturbations with the structure of the convolutional filters (see Sec. 4).

Size of the squares. Given images of size $w \times w$, let $p \in [0, 1]$ be the percentage of elements of x to be modified. The length h of the side of the squares used is given by the closest positive integer to $\sqrt{p \cdot w^2}$ (and $h \geq 3$ for the l_2-attack). Then, the initial p is the only free parameter of our scheme. With $N = 10000$ iterations available, we halve the value of p at $i \in \{10, 50, 200, 1000, 2000, 4000, 6000, 8000\}$ iterations. For different N we rescale the schedule accordingly.

3.3 The l_∞-Square Attack

Initialization. As initialization we use vertical stripes of width one where the color of each stripe is sampled uniformly at random from $\{-\epsilon, \epsilon\}^c$ (c number of color channels). We found that convolutional networks are particularly sensitive to such perturbations, see also [58] for a detailed discussion on the sensitivity of neural networks to various types of high frequency perturbations.

Sampling distribution. Similar to [49] we observe that successful l_∞-perturbations usually have values $\pm\epsilon$ in all the components (note that this does

not hold perfectly due to the image constraints $\hat{x} \in [0,1]^d$). In particular, it holds

$$\hat{x}_i \in \{\max\{0, x_i - \epsilon\}, \min\{1, x_i + \epsilon\}\}.$$

Our sampling distribution P for the l_∞-norm described in Algorithm 2 selects sparse updates of $\hat{x}$ with $\|\delta\|_0 = h \cdot h \cdot c$ where $\delta \in \{-2\epsilon, 0, 2\epsilon\}^d$ and the non-zero elements are grouped to form a square. In this way, after the projection onto the l_∞-ball of radius ϵ (Step 5 of Algorithm 1) all components i for which $\epsilon \leq x_i \leq 1 - \epsilon$ satisfy $\hat{x}_i \in \{x_i - \epsilon, x_i + \epsilon\}$, i.e. differ from the original point x in each element either by ϵ or $-\epsilon$. Thus $\hat{x} - x$ is situated at one of the corners of the l_∞-ball (modulo the components which are close to the boundary). Note that all projections are done by clipping. Moreover, we fix the elements of δ belonging to the same color channel to have the same sign, since we observed that neural networks are particularly sensitive to such perturbations (see Sec. 4.3).

Algorithm 2: Sampling distribution P for l_∞-norm

Input: maximal norm ϵ, window size h, image size w, color channels c

Output: New update δ

$\delta \leftarrow$ array of zeros of size $w \times w \times c$

sample uniformly $r, s \in \{0, \dots, w - h\} \subset \mathbb{N}$

for $i = 1, \dots, c$ **do**

  $\rho \leftarrow Uniform(\{-2\epsilon, 2\epsilon\})$

  $\delta_{r+1:r+h,\, s+1:s+h,\, i} \leftarrow \rho \cdot \mathbb{1}_{h \times h}$

end

3.4 The l_2-Square Attack

Initialization. The l_2-perturbation is initialized by generating a 5×5 grid-like tiling by squares of the image, where the perturbation on each tile has the shape described next in the sampling distribution. The resulting perturbation $\hat{x} - x$ is rescaled to have l_2-norm ϵ and the resulting $\hat{x}$ is projected onto $[0,1]^d$ by clipping.

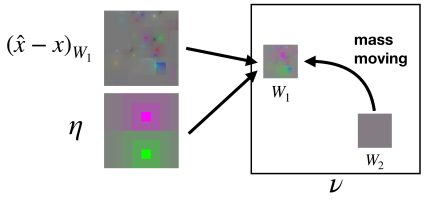

Fig. 2. Perturbation of the l_2-attack

Sampling distribution. First, let us notice that the adversarial perturbations typically found for the l_2-norm tend to be much more localized than those for the l_∞-norm [54], in the sense that large changes are applied on some pixels of the original image, while many others are minimally modified. To mimic this feature we introduce a new update η which has two "centers" with large absolute value and opposite signs, while the other components have lower absolute values as one gets farther away from the centers, but never reaching zero (see Fig. 2 for one example with $h = 8$ of the resulting update η). In this way the modifications are localized and with high contrast between the different halves, which we found to improve the query efficiency. Concretely, we define $\eta^{(h_1, h_2)} \in \mathbb{R}^{h_1 \times h_2}$

Algorithm 3: Sampling distribution P for l_2-norm

Input: maximal norm ϵ, window size h, image size w, number of color channels c, current image $\hat{x}$, original image x

Output: New update δ

$\nu \leftarrow \hat{x} - x$

sample uniformly $r_1, s_1, r_2, s_2 \in \{0, \dots, w-h\}$

$W_1 := r_1 + 1 : r_1 + h, s_1 + 1 : s_1 + h$, $W_2 := r_2 + 1 : r_2 + h, s_2 + 1 : s_2 + h$

$\epsilon^2_{\text{unused}} \leftarrow \epsilon^2 - \|\nu\|_2^2$, $\eta^* \leftarrow \eta / \|\eta\|_2$ with η as in (2)

for $i = 1, \dots, c$ **do**

  $\rho \leftarrow Uniform(\{-1, 1\})$

  $\nu_{\text{temp}} \leftarrow \rho\eta^* + \nu_{W_1,i} / \|\nu_{W_1,i}\|_2$

  $\epsilon^i_{\text{avail}} \leftarrow \sqrt{\|\nu_{W_1 \cup W_2,i}\|_2^2 + \epsilon^2_{\text{unused}}/c}$

  $\nu_{W_2,i} \leftarrow 0$, $\nu_{W_1,i} \leftarrow (\nu_{\text{temp}} / \|\nu_{\text{temp}}\|_2)\epsilon^i_{\text{avail}}$

end

$\delta \leftarrow x + \nu - \hat{x}$

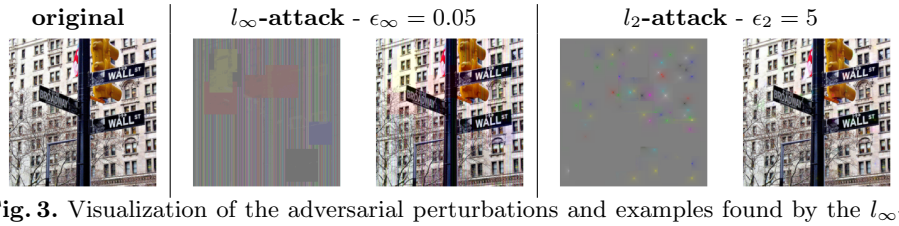

Fig. 3. Visualization of the adversarial perturbations and examples found by the l_∞- and l_2-versions of the Square Attack on ResNet-50

(for some $h_1, h_2 \in \mathbb{N}_+$ such that $h_1 \geq h_2$) for every $1 \leq r \leq h_1$, $1 \leq s \leq h_2$ as

$$\eta^{(h_1,h_2)}_{r,s} = \sum_{k=0}^{M(r,s)} \frac{1}{(n+1-k)^2}, \quad \text{with} \quad n = \left\lfloor \frac{h_1}{2} \right\rfloor,$$

and $M(r,s) = n - \max\{|r - \lfloor \frac{h_1}{2} \rfloor - 1|, |s - \lfloor \frac{h_2}{2} \rfloor - 1|\}$. The intermediate square update $\eta \in \mathbb{R}^{h \times h}$ is then selected uniformly at random from either

$$\eta = \left(\eta^{(h,k)}, -\eta^{(h,h-k)}\right), \quad \text{with } k = \lfloor h/2 \rfloor, \tag{2}$$

or its transpose (corresponding to a rotation of 90°).

Second, unlike l_∞-constraints, l_2-constraints do not allow to perturb each component independently from the others as the overall l_2-norm must be kept smaller than ϵ. Therefore, to modify a perturbation $\hat{x} - x$ of norm ϵ with localized changes while staying on the hypersphere, we have to "move the mass" of $\hat{x} - x$ from one location to another. Thus, our scheme consists in randomly selecting two squared windows in the current perturbation $\nu = \hat{x} - x$, namely ν_{W_1} and ν_{W_2}, setting $\nu_{W_2} = 0$ and using the budget of $\|\nu_{W_2}\|_2$ to increase the total perturbation of ν_{W_1}. Note that the perturbation of W_1 is then a combination of

the existing perturbation plus the new generated η. We report the details of this scheme in Algorithm 3 where step 4 allows to utilize the budget of l_2-norm lost after the projection onto $[0,1]^d$. The update δ output by the algorithm is such that the next iterate $\hat{x}_{\text{new}} = \hat{x} + \delta$ (before projection onto $[0,1]^d$ by clipping) belongs to the hypersphere $B_2(x, \epsilon)$ as stated in the following proposition.

Proposition 1. *Let δ be the output of Algorithm 3. Then $\|\hat{x} + \delta - x\|_2 = \epsilon$.*

4 Theoretical and Empirical Justification of the Method

We provide high-level theoretical justifications and empirical evidence regarding the algorithmic choices in Square Attack, with focus on the l_∞-version (the l_2-version is significantly harder to analyze).

4.1 Convergence Analysis of Random Search

First, we want to study the convergence of the random search algorithm when considering an L-smooth objective function g (such as neural networks with activation functions like softplus, swish, ELU, etc) on the whole space $\mathbb{R}^d$ (without projection[4]) under the following assumptions on the update δ_t drawn from the sampling distribution P_t:

$$\mathbb{E}\|\delta_t\|_2^2 \leq \gamma_t^2 C \text{ and } \mathbb{E}|\langle \delta_t, v\rangle| \geq \tilde{C}\gamma_t \|v\|_2, \ \forall v \in \mathbb{R}^d, \tag{3}$$

where γ_t is the step size at iteration t, $C, \tilde{C} > 0$ some constants and $\langle \cdot, \cdot \rangle$ denotes the inner product. We obtain the following result, similar to existing convergence rates for zeroth-order methods [42,43,24]:

Proposition 2. *Suppose that $\mathbb{E}[\delta_t] = 0$ and the assumptions in Eq. (3) hold. Then for step-sizes $\gamma_t = \gamma/\sqrt{T}$, we have*

$$\min_{t=0,\dots,T} \mathbb{E}\|\nabla g(x_t)\|_2 \leq \frac{2}{\gamma \tilde{C}\sqrt{T}} \left(g(x_0) - \mathbb{E}g(x_{T+1}) + \frac{\gamma^2 C L}{2} \right).$$

This basically shows for T large enough one can make the gradient arbitrary small, meaning that the random search algorithm converges to a critical point of g (one cannot hope for much stronger results in non-convex optimization without stronger conditions).

Unfortunately, the second assumption in Eq. (3) does not directly hold for our sampling distribution P for the l_∞-norm (see Sup. A.3), but holds for a similar one, P^{multiple}, where each component of the update δ is drawn uniformly at random from $\{-2\epsilon, 2\epsilon\}$. In fact we show in Sup. A.4, using the Khintchine inequality [29], that

$$\mathbb{E}\|\delta_t\|_2^2 \leq 4c\varepsilon^2 h^2 \text{ and } \mathbb{E}|\langle \delta_t, v\rangle| \geq \frac{\sqrt{2}c\varepsilon h^2}{d}\|v\|_2, \ \forall v \in \mathbb{R}^d.$$

Moreover, while P^{multiple} performs worse than the distribution used in Algorithm 2, we show in Sec. 4.3 that it already reaches state-of-the-art results.

[4] Nonconvex constrained optimization under noisy oracles is notoriously harder [22].

4.2 Why Squares?

Previous works [49,39] build their l_∞-attacks by iteratively adding square modifications. Likewise we change square-shaped regions of the image for both our l_∞- and l_2-attacks—with the difference that we can sample any square subset of the input, while the grid of the possible squares is fixed in [49,39]. This leads naturally to wonder why squares are superior to other shapes, e.g. rectangles.

Let us consider the l_∞-threat model, with bound ϵ, input space $\mathbb{R}^{d\times d}$ and a convolutional filter $w \in \mathbb{R}^{s\times s}$ with entries unknown to the attacker. Let $\delta \in \mathbb{R}^{d\times d}$ be the sparse update with $\|\delta\|_0 = k \geq s^2$ and $\|\delta\|_\infty \leq \epsilon$. We denote by $S(a,b)$ the index set of the rectangular support of δ with $|S(a,b)| = k$ and shape $a \times b$. We want to provide intuition why sparse square-shaped updates are superior to rectangular ones in the sense of reaching a maximal change in the activations of the first convolutional layer.

Let $z = \delta * w \in \mathbb{R}^{d\times d}$ denote the output of the convolutional layer for the update δ. The l_∞-norm of z is the maximal componentwise change of the convolutional layer:

$$\begin{aligned}\|z\|_\infty &= \max_{u,v} |z_{u,v}| = \max_{u,v} \Big| \sum_{i,j=1}^{s} \delta_{u-\lfloor \frac{s}{2} \rfloor + i, v-\lfloor \frac{s}{2} \rfloor + j} \cdot w_{i,j} \Big| \\ &\leq \max_{u,v} \epsilon \sum_{i,j} |w_{i,j}| \mathbb{1}_{(u-\lfloor \frac{s}{2} \rfloor + i, v-\lfloor \frac{s}{2} \rfloor + j) \in S(a,b)},\end{aligned}$$

where elements with indices exceeding the size of the matrix are set to zero. Note that the indicator function attains 1 only for the non-zero elements of δ involved in the convolution to get $z_{u,v}$. Thus, to have the largest upper bound possible on $|z_{u,v}|$, for some (u,v), we need the largest possible amount of components of δ with indices in

$$C(u,v) = \Big\{ (u - \lfloor \frac{s}{2} \rfloor + i, v - \lfloor \frac{s}{2} \rfloor + j) : \ i,j = 1, \dots, s \Big\}$$

to be non-zero (that is in $S(a,b)$).

Therefore, it is desirable to have the shape $S(a,b)$ of the perturbation δ selected so to maximize the number N of convolutional filters $w \in \mathbb{R}^{s\times s}$ which fit into the rectangle $a \times b$. Let $\mathcal{F}$ be the family of the objects that can be defined as the union of axis-aligned rectangles with vertices on $\mathbb{N}^2$, and $\mathcal{G} \subset \mathcal{F}$ be the squares of $\mathcal{F}$ of shape $s \times s$ with $s \geq 2$. We have the following proposition:

Proposition 3. *Among the elements of $\mathcal{F}$ with area $k \geq s^2$, those which contain the largest number of elements of $\mathcal{G}$ have*

$$N^* = (a - s + 1)(b - s + 1) + (r - s + 1)^+ \tag{4}$$

of them, with $a = \Big\lfloor \sqrt{k} \Big\rfloor$, $b = \lfloor \frac{k}{a} \rfloor$, $r = k - ab$ and $z^+ = \max\{z, 0\}$.

This proposition states that, if we can modify only k elements of δ, then shaping them to form (approximately) a square allows to maximize the number of pairs (u,v) for which $|S(a,b) \cap C(u,v)| = s^2$. If $k = l^2$ then $a = b = l$ are the optimal values for the shape of the perturbation update, i.e. the shape is exactly a square.

Table 1. Ablation study of the l_∞-Square Attack which shows how the individual design decisions improve the performance. The fourth row corresponds to the method for which we have shown convergence guarantees in Sec. 4.1. The last row corresponds to our final l_∞-attack. c indicates the number of color channels, h the length of the side of the squares, so that "# random sign" c represents updates with constant sign for each color, while $c \cdot h^2$ updates with signs sampled independently of each other

Update shape	# random signs	Initialization	Failure rate	Avg. queries	Median queries
random	$c \cdot h^2$	vert. stripes	**0.0%**	401	48
random	$c \cdot h^2$	uniform rand.	**0.0%**	393	132
random	c	vert. stripes	**0.0%**	339	53
square	$c \cdot h^2$	vert. stripes	**0.0%**	153	15
rectangle	c	vert. stripes	**0.0%**	93	16
square	c	uniform rand.	**0.0%**	91	26
square	c	vert. stripes	**0.0%**	**73**	**11**

4.3 Ablation Study

We perform an ablation study to show how the individual design decisions for the sampling distribution of the random search improve the performance of l_∞-Square Attack, confirming the theoretical arguments above. The comparison is done for an l_∞-threat model of radius $\epsilon = 0.05$ on $1,000$ test points for a ResNet-50 model trained normally on ImageNet (see Sec. 5 for details) with a query limit of $10,000$ and results are shown in Table 1. Our sampling distribution is special in two aspects: i) we use localized update shapes in form of squares and ii) the update is constant in each color channel. First, one can observe that our update shape "square" performs better than "rectangle" as we discussed in the previous section, and it is significantly better than "random" (the same amount of pixels is perturbed, but selected randomly in the image). This holds both for c (constant sign per color channel) and $c \cdot h^2$ (every pixel and color channel is changed independently of each other), with an improvement in terms of average queries of 339 to 73 and 401 to 153 respectively. Moreover, with updates of the same shape, the constant sign over color channels is better than selecting it uniformly at random (improvement in average queries: 401 to 339 and 153 to 73). In total the algorithm with "square-c" needs more than $5\times$ less average queries than "random-$c \cdot h^2$", showing that our sampling distribution is the key to the high query efficiency of Square Attack.

The last innovation of our random search scheme is the initialization, crucial element of every non-convex optimization algorithm. Our method ("square-c") with the vertical stripes initialization improves over a uniform initialization on average by $\approx 25\%$ and, especially, median number of queries (more than halved).

We want to also highlight that the sampling distribution "square-$c \cdot h^2$" for which we shown convergence guarantees in Sec. 4.1 performs already better in terms of the success rate and the median number of queries than the state of the art (see Sec. 5). For a more detailed ablation, also for our l_2-attack, see Sup. C.

Table 2. Results of **untargeted** attacks on ImageNet with a limit of 10,000 queries. For the l_∞-attack we set the norm bound $\epsilon = 0.05$ and for the l_2-attack $\epsilon = 5$. Models: normally trained **I**: Inception v3, **R**: ResNet-50, **V**: VGG-16-BN. The Square Attack outperforms for both threat models all other methods in terms of success rate and query efficiency. The missing entries correspond to the results taken from the original paper where some models were not reported

Norm	Attack	Failure rate			Avg. queries			Med. queries		
		I	R	V	I	R	V	I	R	V
l_∞	Bandits [31]	3.4%	1.4%	2.0%	957	727	394	218	136	36
	Parsimonious [49]	1.5%	-	-	722	-	-	237	-	-
	DFO_c–CMA [39]	0.8%	**0.0%**	0.1%	630	270	219	259	143	107
	DFO_d–Diag. CMA [39]	2.3%	1.2%	0.5%	424	417	211	**20**	20	2
	SignHunter [2]	1.0%	0.1%	0.3%	471	129	95	95	39	43
	Square Attack	**0.3%**	**0.0%**	**0.0%**	**197**	**73**	**31**	24	**11**	**1**
l_2	Bandits [31]	9.8%	6.8%	10.2%	1486	939	511	660	392	196
	SimBA-DCT [28]	35.5%	12.7%	7.9%	**651**	**582**	452	564	467	360
	Square Attack	**7.1%**	**0.7%**	**0.8%**	1100	616	**377**	**385**	**170**	**109**

5 Experiments

In this section we show the effectiveness of the Square Attack. Here we concentrate on **untargeted** attacks since our primary goal is query efficient robustness evaluation, while the **targeted** attacks are postponed to the supplement. First, we follow the standard setup [31,39] of comparing black-box attacks on three ImageNet models in terms of success rate and query efficiency for the l_∞- and l_2-untargeted attacks (Sec. 5.1). Second, we show that our *black-box* attack can even outperform *white-box* PGD attacks on several models (Sec. 5.2). Finally, in the supplement we provide more experimental details (Sup. B), a stability study of our attack for different parameters (Sup. C) and random seeds (Sup. D), and additional results including the experiments for targeted attacks (Sup. E).

5.1 Evaluation on ImageNet

We compare the Square Attack to state-of-the-art score-based black-box attacks (without any extra information such as surrogate models) on three pretrained models in PyTorch (Inception v3, ResNet-50, VGG-16-BN) using 1,000 images from the ImageNet validation set. Unless mentioned otherwise, we use the code from the other papers with their suggested parameters. As it is standard in the literature, we give a budget of 10,000 queries per point to find an adversarial perturbation of l_p-norm at most ϵ. We report the *average* and *median* number of queries each attack requires to craft an adversarial example, together with the *failure rate*. All query statistics are computed only for successful attacks on the points which were originally correctly classified.

Tables 2 and 3 show that the Square Attack, despite its simplicity, achieves in all the cases (models and norms) the **lowest failure rate**, ($< 1\%$ everywhere

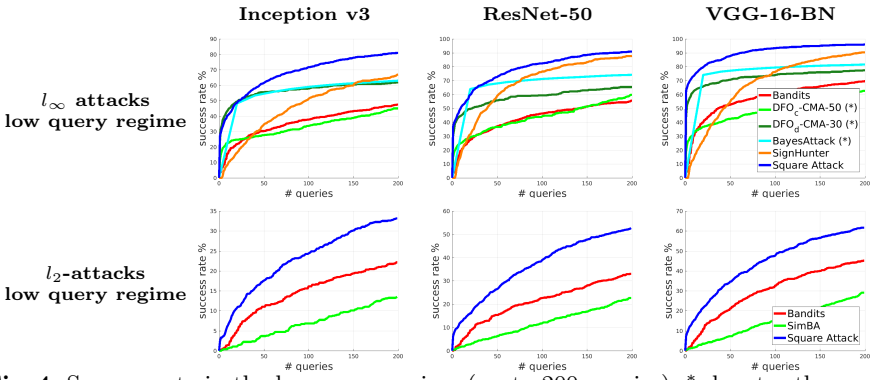

Fig. 4. Success rate in the low-query regime (up to 200 queries). * denotes the results obtained via personal communication with the authors and evaluated on 500 and 10,000 randomly sampled points for BayesAttack [50] and DFO [39] methods, respectively

except for the l_2-attack on Inception v3), and almost always requires **fewer queries** than the competitors to succeed. Fig. 4 shows the progression of the success rate of the attacks over the first 200 queries. Even in the low query regime the Square Attack outperforms the competitors for both norms. Finally, we highlight that the only hyperparameter of our attack, p, regulating the size of the squares, is set for all the models to 0.05 for l_∞ and 0.1 for l_2-perturbations.

l_∞-attacks. We compare our attack to Bandits [32], Parsimonious [49], DFO$_c$ / DFO$_d$ [39], and SignHunter [2]. In Table 2 we report the results of the l_∞-attacks with norm bound of $\epsilon = 0.05$. The Square Attack always has the lowest failure rate, notably 0.0% in 2 out of 3 cases, and the lowest query consumption. Interestingly, our attack has median equal 1 on VGG-16-BN, meaning that the proposed initialization is particularly effective for this model.

The closest competitor in terms of the *average* number of queries is SignHunter [2], which still needs on average between 1.8 and 3 times more queries to find adversarial examples and has a higher failure rate than our attack. Moreover, the median number of queries of SignHunter is much worse than for our method (e.g. 43 vs 1 on VGG). We note that although DFO$_c$–CMA [39] is competitive to our attack in terms of *median* queries, it has a significantly higher failure rate and between 2 and 7 times worse average number of queries. Additionally, our method is also more effective in the low-query regime (Fig. 4) than other methods (including [50]) on all the models.

Table 3. Query statistics for untargeted l_2-attacks computed for the points for which all three attacks are successful for fair comparison

Attack	Avg. queries			Med. queries		
	I	R	V	I	R	V
Bandits [31]	536	635	398	368	314	177
SimBA-DCT [28]	647	563	421	552	446	332
Square Attack	**352**	**287**	**217**	**181**	**116**	**80**

l_2-attacks. We compare our attack to Bandits [31] and SimBA [28] for $\epsilon = 5$, while we do not consider SignHunter [2] since it is not as competitive as for the l_∞-norm, and in particular worse than Bandits on ImageNet (see Fig. 2 in [2]).

As Table 2 and Fig. 4 show, the Square Attack outperforms by a large margin the other methods in terms of failure rate, and achieves the lowest median number of queries for all the models and the lowest average one for VGG-16-BN. However, since it has a significantly lower failure rate, the statistics of the Square Attack are biased by the "hard" cases where the competitors fail. Then, we recompute the same statistics considering only the points where all the attacks are successful (Table 3). In this case, our method improves by at least 1.5× the average and by at least 2× the median number of queries.

5.2 Square Attack Can be More Accurate than White-box Attacks

Here we test our attack on problems which are challenging for both white-box PGD and other black-box attacks. We use for evaluation *robust accuracy*, defined as the worst-case accuracy of a classifier when an attack perturbs each input in some l_p-ball. We show that our algorithm outperforms the competitors both on state-of-the-art robust models and defenses that induce different types of gradient masking. Thus, our attack is useful to evaluate robustness without introducing adaptive attacks designed for each model separately.

Table 4. On the robust models of [37] and [61] on MNIST l_∞-Square Attack with $\epsilon = 0.3$ achieves state-of-the-art (SOTA) results outperforming white-box attacks

Model	**Robust accuracy**	
	SOTA	**Square**
Madry et al. [37]	**88.13%**	88.25%
TRADES [61]	93.33%	**92.58%**

Outperforming white-box attacks on robust models. The models obtained with the adversarial training of [37] and TRADES [61] are standard benchmarks to test adversarial attacks, which means that many papers have tried to reduce their robust accuracy, without limit on the computational budget and primarily via white-box attacks. We test our l_∞-Square Attack on these robust models on MNIST at $\epsilon = 0.3$, using $p = 0.8$, 20k queries and 50 random restarts, i.e., we run our attack 50 times and consider it successful if any of the runs finds an adversarial example (Table 4). On the model of Madry et al [37] Square Attack is only 0.12% far from the *white-box* state-of-the-art, achieving the second best result (also outperforming the 91.47% of SignHunter [2] by a large margin). On the TRADES benchmark [63], our method obtains a new SOTA of 92.58% robust accuracy outperforming the white-box attack of [20]. Additionally, the subsequent work of [21] uses the Square Attack as part of their *AutoAttack* where they show that the Square Attack outperforms other white-box attacks on 9 out of 9 MNIST models they evaluated. Thus, our black-box attack can be also useful for robustness evaluation of new defenses in the setting where gradient-based attacks require many restarts and iterations.

Resistance to gradient masking. In Table 5 we report the robust accuracy at different thresholds ϵ of the l_∞-adversarially trained models on MNIST of [37] for the l_2-threat model. It is known that the PGD is ineffective since it suffers from gradient masking [53]. Unlike PGD and other black-box attacks, our Square Attack does not suffer from gradient masking and yields robust accuracy close to zero for $\epsilon = 2.5$, with only a single run. Moreover, the l_2-version of SignHunter [2]

Table 5. l_2-robustness of the l_∞-adversarially trained models of [37] at different thresholds ϵ. PGD is shown with 1, 10, 100 random restarts. The black-box attacks are given a 10k queries budget (see the supplement for details)

ϵ_2	Robust accuracy						
	White-box			Black-box			
	PGD_1	PGD_{10}	PGD_{100}	SignHunter	Bandits	SimBA	**Square**
2.0	79.6%	67.4%	**59.8%**	95.9%	80.1%	87.6%	**16.7%**
2.5	69.2%	51.3%	**36.0%**	94.9%	32.4%	75.8%	**2.4%**
3.0	57.6%	29.8%	**12.7%**	93.8%	12.5%	58.1%	**0.6%**

Table 6. l_∞-robustness of Clean Logit Pairing (CLP), Logit Squeezing (LSQ) [33]. The Square Attack is competitive to white-box PGD with many restarts (R=10,000, R=100 on MNIST, CIFAR-10 resp.) and more effective than black-box attacks [31,2]

ϵ_∞	Model	Robust accuracy				
		White-box		Black-box		
		PGD_1	PGD_R	Bandits	SignHunter	**Square**
0.3	CLP_{MNIST}	62.4%	**4.1%**	33.3%	62.1%	**6.1%**
	LSQ_{MNIST}	70.6%	**5.0%**	37.3%	65.7%	**2.6%**
16/255	CLP_{CIFAR}	2.8%	**0.0%**	14.3%	**0.1%**	0.2%
	LSQ_{CIFAR}	27.0%	**1.7%**	27.7%	13.2%	**7.2%**

fails to accurately assess the robustness because the method optimizes only over the extreme points of the l_∞-ball of radius $\epsilon/\sqrt{d}$ embedded in the target l_2-ball.

Attacking Clean Logit Pairing and Logit Squeezing. These two l_∞ defenses proposed in [33] were broken in [40]. However, [40] needed up to 10k restarts of PGD which is computationally prohibitive. Using the publicly available models from [40], we run the Square Attack with $p = 0.3$ and 20k query limit (results in Table 6). We obtain robust accuracy similar to PGD_R in most cases, but with a *single run*, i.e. without additional restarts. At the same time, although on some models Bandits and SignHunter outperform PGD_1, they on average achieve significantly worse results than the Square Attack. This again shows the utility of the Square Attack to accurately assess robustness.

6 Conclusion

We have presented a simple black-box attack which outperforms by a large margin the state-of-the-art both in terms of query efficiency and success rate. Our results suggest that our attack is useful *even in comparison to white-box attacks* to better estimate the robustness of models that exhibit gradient masking.

Acknowledgements. We thank L. Meunier and S. N. Shukla for providing the data for Figure 6. M.A. thanks A. Modas for fruitful discussions. M.H and F.C. acknowledge support by the Tue.AI Center (FKZ: 01IS18039A), DFG TRR 248, project number 389792660 and DFG EXC 2064/1, project number 390727645.

References

1. Akhtar, N., Mian, A.: Threat of adversarial attacks on deep learning in computer vision: A survey. IEEE Access **6**, 14410–14430 (2018)
2. Al-Dujaili, A., O'Reilly, U.M.: There are no bit parts for sign bits in black-box attacks. In: ICLR (2020)
3. Alzantot, M., Sharma, Y., Chakraborty, S., Srivastava, M.: Genattack: practical black-box attacks with gradient-free optimization. In: Genetic and Evolutionary Computation Conference (GECCO) (2019)
4. Athalye, A., Carlini, N., Wagner, D.A.: Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In: ICML (2018)
5. Bastani, O., Ioannou, Y., Lampropoulos, L., Vytiniotis, D., Nori, A., Criminisi, A.: Measuring neural net robustness with constraints. In: NeurIPS (2016)
6. Bhagoji, A.N., He, W., Li, B., Song, D.: Practical black-box attacks on deep neural networks using efficient query mechanisms. In: ECCV (2018)
7. Biggio, B., Roli, F.: Wild patterns: Ten years after the rise of adversarial machine learning. Pattern Recognition **84**, 317–331 (2018)
8. Boyd, S., Vandenberghe, L.: Convex Optimization. Cambridge University Press, Cambridge (2004)
9. Brendel, W., Rauber, J., Bethge, M.: Decision-based adversarial attacks: Reliable attacks against black-box machine learning models. In: ICLR (2018)
10. Brown, T.B., Mané, D., Roy, A., Abadi, M., Gilmer, J.: Adversarial patch. In: NeurIPS 2017 Workshop on Machine Learning and Computer Security (2017)
11. Brunner, T., Diehl, F., Le, M.T., Knoll, A.: Guessing smart: biased sampling for efficient black-box adversarial attacks. In: ICCV (2019)
12. Carlini, N., Wagner, D.: Adversarial examples are not easily detected: Bypassing ten detection methods. In: ACM Workshop on Artificial Intelligence and Security (2017)
13. Chen, J., Jordan, M.I., J., W.M.: HopSkipJumpAttack: a query-efficient decision-based attack (2019), arXiv preprint arXiv:1904.02144
14. Chen, P., Sharma, Y., Zhang, H., Yi, J., Hsieh, C.: Ead: Elastic-net attacks to deep neural networks via adversarial examples. In: AAAI (2018)
15. Chen, P.Y., Zhang, H., Sharma, Y., Yi, J., Hsieh, C.J.: Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In: 10th ACM Workshop on Artificial Intelligence and Security - AISec '17. ACM Press (2017)
16. Cheng, M., Le, T., Chen, P.Y., Yi, J., Zhang, H., Hsieh, C.J.: Query-efficient hard-label black-box attack: An optimization-based approach. In: ICLR (2019)
17. Cheng, S., Dong, Y., Pang, T., Su, H., Zhu, J.: Improving black-box adversarial attacks with a transfer-based prior. In: NeurIPS (2019)
18. Cohen, J.M., Rosenfeld, E., Kolter, Z.: Certified adversarial robustness via randomized smoothing. In: ICML (2019)
19. Croce, F., Hein, M.: Sparse and imperceivable adversarial attacks. In: ICCV (2019)
20. Croce, F., Hein, M.: Minimally distorted adversarial examples with a fast adaptive boundary attack. In: ICML (2020)
21. Croce, F., Hein, M.: Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In: ICML (2020)
22. Davis, D., Drusvyatskiy, D.: Stochastic model-based minimization of weakly convex functions. SIAM Journal on Optimization **29**(1), 207–239 (2019)

23. Du, J., Zhang, H., Zhou, J.T., Yang, Y., Feng, J.: Query-efficient meta attack to deep neural networks. In: ICLR (2020)
24. Duchi, J., Jordan, M., Wainwright, M., Wibisono, A.: Optimal rates for zero-order convex optimization: The power of two function evaluations. IEEE Transactions on Information Theory **61**(5), 2788–2806 (2015)
25. Fawzi, A., Frossard, P.: Measuring the effect of nuisance variables on classifiers. In: British Machine Vision Conference (BMVC) (2016)
26. Gu, S., Rigazio, L.: Towards deep neural network architectures robust to adversarial examples. In: ICLR Workshop (2015)
27. Guo, C., Frank, J.S., Weinberger, K.Q.: Low frequency adversarial perturbation. In: UAI (2019)
28. Guo, C., Gardner, J.R., You, Y., Wilson, A.G., Weinberger, K.Q.: Simple black-box adversarial attacks. In: ICML (2019)
29. Haagerup, U.: The best constants in the Khintchine inequality. Studia Math. **70**(3), 231–283 (1981)
30. Ilyas, A., Engstrom, L., Athalye, A., Lin, J.: Black-box adversarial attacks with limited queries and information. In: ICML (2018)
31. Ilyas, A., Engstrom, L., Madry, A.: Prior convictions: Black-box adversarial attacks with bandits and priors. In: ICLR (2019)
32. Ilyas, A., Santurkar, S., Tsipras, D., Engstrom, L., Tran, B., Madry, A.: Adversarial examples are not bugs, they are features. NeurIPS (2019)
33. Kannan, H., Kurakin, A., Goodfellow, I.: Adversarial logit pairing (2018), arXiv preprint arXiv:1803.06373
34. Karmon, D., Zoran, D., Goldberg, Y.: Lavan: Localized and visible adversarial noise. In: ICML (2018)
35. Li, Y., Li, L., Wang, L., Zhang, T., Gong, B.: Nattack: Learning the distributions of adversarial examples for an improved black-box attack on deep neural networks. In: ICML (2019)
36. Lin, Y., Jiang, H., Jiang, H.: Bandlimiting neural networks against adversarial attacks (2019), arXiv preprint arXiv:1905.12797
37. Madry, A., Makelov, A., Schmidt, L., Tsipras, D., Vladu, A.: Towards deep learning models resistant to adversarial attacks. In: ICLR (2018)
38. Matyas, J.: Random optimization. Automation and Remote control **26**(2), 246–253 (1965)
39. Meunier, L., Atif, J., Teytaud, O.: Yet another but more efficient black-box adversarial attack: tiling and evolution strategies (2019), arXiv preprint, arXiv:1910.02244
40. Mosbach, M., Andriushchenko, M., Trost, T., Hein, M., Klakow, D.: Logit pairing methods can fool gradient-based attacks. In: NeurIPS 2018 Workshop on Security in Machine Learning (2018)
41. Narodytska, N., Kasiviswanathan, S.: Simple black-box adversarial attacks on deep neural networks. In: CVPR Workshops (2017)
42. Nemirovsky, A.S., Yudin, D.B.: Problem Complexity and Method Efficiency in Optimization. Wiley-Interscience Series in Discrete Mathematics, John Wiley & Sons (1983)
43. Nesterov, Y., Spokoiny, V.: Random gradient-free minimization of convex functions. Foundations of Computational Mathematics **17**(2), 527–566 (2017)
44. Papernot, N., McDaniel, P., Goodfellow, I.: Transferability in machine learning: from phenomena to black-box attacks using adversarial samples (2016), arXiv preprint arXiv:1605.07277

45. Papernot, N., McDaniel, P., Wu, X., Jha, S., Swami, A.: Distillation as a defense to adversarial perturbations against deep networks. In: IEEE Symposium on Security & Privacy (2016)
46. Rastrigin, L.: The convergence of the random search method in the extremal control of a many parameter system. Automaton & Remote Control **24**, 1337–1342 (1963)
47. Schrack, G., Choit, M.: Optimized relative step size random searches. Mathematical Programming **10**, 230–244 (1976)
48. Schumer, M., Steiglitz, K.: Adaptive step size random search. IEEE Transactions on Automatic Control **13**(3), 270–276 (1968)
49. Seungyong, M., Gaon, A., Hyun, O.S.: Parsimonious black-box adversarial attacks via efficient combinatorial optimization. In: ICML (2019)
50. Shukla, S.N., Sahu, A.K., Willmott, D., Kolter, Z.: Black-box adversarial attacks with Bayesian optimization (2019), arXiv preprint arXiv:1909.13857
51. Su, J., Vargas, D., Sakurai, K.: One pixel attack for fooling deep neural networks. IEEE Transactions on Evolutionary Computation (2019)
52. Suya, F., Chi, J., Evans, D., Tian, Y.: Hybrid batch attacks: Finding black-box adversarial examples with limited queries (2019), arXiv preprint, arXiv:1908.07000
53. Tramèr, F., Boneh, D.: Adversarial training and robustness for multiple perturbations. In: NeurIPS (2019)
54. Tsipras, D., Santurkar, S., Engstrom, L., Turner, A., Madry, A.: Robustness may be at odds with accuracy. In: ICLR (2019)
55. Tu, C.C., Ting, P., Chen, P.Y., Liu, S., Zhang, H., Yi, J., Hsieh, C.J., Cheng, S.M.: Autozoom: Autoencoder-based zeroth order optimization method for attacking black-box neural networks. In: AAAI Conference on Artificial Intelligence (2019)
56. Uesato, J., O'Donoghue, B., Van den Oord, A., Kohli, P.: Adversarial risk and the dangers of evaluating against weak attacks. In: ICML (2018)
57. Yan, Z., Guo, Y., Zhang, C.: Subspace attack: Exploiting promising subspaces for query-efficient black-box attacks. In: NeurIPS (2019)
58. Yin, D., Lopes, R.G., Shlens, J., Cubuk, E.D., Gilmer, J.: A Fourier perspective on model robustness in computer vision. In: NeurIPS (2019)
59. Yu, F., Koltun, V., Funkhouser, T.: Dilated residual networks. In: CVPR (2017)
60. Zabinsky, Z.B.: Random search algorithms. Wiley encyclopedia of operations research and management science (2010)
61. Zhang, H., Yu, Y., Jiao, J., Xing, E.P., Ghaoui, L.E., Jordan, M.I.: Theoretically principled trade-off between robustness and accuracy. In: ICML (2019)
62. Zheng, S., Song, Y., Leung, T., Goodfellow, I.J.: Improving the robustness of deep neural networks via stability training. In: CVPR (2016)
63. Zheng, T., Chen, C., Ren, K.: Distributionally adversarial attack. In: AAAI (2019)

Supplementary Material

Organization of the Supplementary Material

In Section A, we present the missing proofs of Section 3 and Section 4 and slightly deepen our theoretical insights on the efficiency of the proposed l_∞-attack. Section B covers various implementation details and the hyperparameters we used. We show a more detailed ablation study on different choices of the attack's algorithm in Section C. Since the Square Attack is a randomized algorithm, we show the variance of the main reported metric for different random seeds in Section D. Finally, Section E presents results of the targeted attacks on ImageNet, additional results for the untargeted attacks, and an evaluation of the post-averaging defense [36] which we conclude is much less robust than claimed.

A Proofs Omitted from Section 3 and Section 4

In this section, we present the proofs omitted from Section 3 and Section 4.

A.1 Proof of Proposition 1

Let δ be the output of Algorithm 3. We prove here that $\|\hat{x} + \delta - x\|_2 = \epsilon$.

From Step 13 of Algorithm 3, we directly have the equality $\|\hat{x} + \delta - x\|_2 = \|\nu\|_2$. Let ν^{old} be the update at the previous iteration, defined in Step 1 and $\overline{W_1 \cup W_2}$ the indices not belonging to $W_1 \cup W_2$. Then,

$$
\begin{aligned}
\|\nu\|_2^2 &= \sum_{i=1}^{c} \|\nu_{W_1 \cup W_2, i}\|_2^2 + \sum_{i=1}^{c} \left\|\nu_{\overline{W_1 \cup W_2}, i}\right\|_2^2 \\
&= \sum_{i=1}^{c} \|\nu_{W_1, i}\|_2^2 + \sum_{i=1}^{c} \left\|\nu_{\overline{W_1 \cup W_2}, i}\right\|_2^2 \\
&= \sum_{i=1}^{c} (\epsilon^i_{\text{avail}})^2 + \sum_{i=1}^{c} \left\|\nu_{\overline{W_1 \cup W_2}, i}\right\|_2^2 \\
&= \sum_{i=1}^{c} \left\|\nu^{\text{old}}_{W_1 \cup W_2, i}\right\|_2^2 + \epsilon^2_{\text{unused}} + \sum_{i=1}^{c} \left\|\nu_{\overline{W_1 \cup W_2}, i}\right\|_2^2 \\
&\overset{(i)}{=} \sum_{i=1}^{c} \left\|\nu^{\text{old}}_{W_1 \cup W_2, i}\right\|_2^2 + \epsilon^2_{\text{unused}} + \sum_{i=1}^{c} \left\|\nu^{\text{old}}_{\overline{W_1 \cup W_2}, i}\right\|_2^2 \\
&= \left\|\nu^{\text{old}}\right\|_2^2 + \epsilon^2_{\text{unused}} \overset{(ii)}{=} \epsilon^2,
\end{aligned}
$$

where (i) holds since $\nu^{\text{old}}_{\overline{W_1 \cup W_2}} \equiv \nu_{\overline{W_1 \cup W_2}}$ as the modifications affect only the elements in the two windows, and (ii) holds by the definition of ϵ_{unused} in Step 4 of Algorithm 3.

A.2 Proof of Proposition 2

Using the L-smoothness of the function g, that is it holds for all $x, y \in \mathbb{R}^d$,

$$\|\nabla g(x) - \nabla g(y)\|_2 \leq L \|x - y\|_2 .$$

we obtain (see e.g. [8]):

$$g(x_t + \delta_t) \leq g(x_t) + \langle \nabla g(x_t), \delta_t \rangle + \frac{L}{2}\|\delta_t\|_2^2,$$

and by definition of x_{t+1} we have

$$\begin{aligned} g(x_{t+1}) &\leq \min\{g(x_t), g(x_t + \delta_t)\} \\ &\leq g(x_t) + \min\{0, \langle \nabla g(x_t), \delta_t \rangle + \frac{L}{2}\|\delta_t\|_2^2\}. \end{aligned}$$

Using the definition of the min as a function of the absolute value ($2\min\{a, b\} = a + b - |a - b|$) yields

$$g(x_{t+1}) \leq g(x_t) + \frac{1}{2}\langle \nabla g(x_t), \delta_t \rangle + \frac{L}{4}\|\delta_t\|_2^2 - \frac{1}{2}|\langle \nabla g(x_t), \delta_t \rangle + \frac{L}{2}\|\delta_t\|_2^2|.$$

And using the triangular inequality ($|a + b| \geq |a| - |b|$), we have

$$g(x_{t+1}) \leq g(x_t) + \frac{1}{2}\langle \nabla g(x_t), \delta_t \rangle + \frac{L}{2}\|\delta_t\|_2^2 - \frac{1}{2}|\langle \nabla g(x_t), \delta_t \rangle|.$$

Therefore taking the expectation and using that $\mathbb{E}\delta_t = 0$, we get

$$\mathbb{E}g(x_{t+1}) \leq \mathbb{E}g(x_t) - \frac{1}{2}\mathbb{E}|\langle \nabla g(x_t), \delta_t \rangle| + \frac{L}{2}\mathbb{E}\|\delta_t\|_2^2.$$

Therefore, together with the assumptions in Eq. (3) this yields to

$$\mathbb{E}g(x_{t+1}) \leq \mathbb{E}g(x_t) - \frac{\tilde{C}\gamma_t}{2}\mathbb{E}\|\nabla g(x_t)\|_2 + \frac{LC\gamma_t^2}{2}.$$

and thus

$$\mathbb{E}\,\|\nabla g(x_t)\|_2 \leq \frac{2}{\gamma_t \tilde{C}}\Big(\mathbb{E}g(x_t) - \mathbb{E}g(x_{t+1}) + \frac{LC\gamma_t^2}{2}\Big).$$

Thus for $\gamma_t = \gamma$ we have summing for $t = 0 : T$

$$\begin{aligned} \min_{0 \leq i \leq T} \mathbb{E}\|\nabla g(x_i)\|_2 &\leq \frac{1}{T}\sum_{t=0}^{T} \mathbb{E}\|\nabla g(x_t)\|_2 \\ &\leq \frac{2}{\tilde{C}\gamma T}\big[g(x_0) - \mathbb{E}g(x_{T+1}) + \frac{TLC\gamma^2}{2}\big]. \end{aligned}$$

We conclude setting the step-size to $\gamma = \Theta(1/\sqrt{T})$.

A.3 Assumptions in Eq. (3) Do Not Hold for the Sampling Distribution P

Let us consider an update δ with a window size $h = 2$ and the direction $v \in \{-1, 1\}^{w \times w \times c}$ defined as

$$v^i_{k,l} = (-1)^{kl} \quad \text{for all } i, k, l.$$

It is easy to check that any update δ drawn from the sampling distribution P is orthogonal to this direction v:

$$\langle v, \delta \rangle = \sum_{i=1}^{c} \sum_{k=r+1}^{r+2} \sum_{l=s+1}^{s+2} (-1)^{kl} = c(-1 + 1 - 1 + 1) = 0.$$

Thus, $\mathbb{E}|\langle v, \delta \rangle| = 0$ and the assumptions in Eq. (3) do not hold. This means that the convergence analysis does not directly hold for the sampling distribution P.

A.4 Assumptions in Eq. (3) Hold for the Sampling Distribution P^{multiple}

Let us consider the sampling distribution P^{multiple} where different Rademacher $\rho_{k,l,i}$ are drawn for each pixel of the update window $\delta_{r+1:r+h,\, s+1:s+h,\, i}$. We present it in Algorithm 4 with the convention that any subscript $k > w$ should be understood as $k - w$. This technical modification is greatly helpful to avoid side effect.

Let $v \in \mathbb{R}^{w \times w \times c}$ for which we have using the Khintchine inequality [29]:

Algorithm 4: Sampling distribution P^{multiple} for l_∞-norm

Input: maximal norm ϵ, window size h, image size w, color channels c
Output: New update δ

$\delta \leftarrow$ array of zeros of size $w \times w \times c$
sample uniformly $r, s \in \{0, \dots, w\} \subset \mathbb{N}$
for $i = 1, \dots, c$ **do**
  $\delta_{r+1:r+h,\, s+1:s+h,\, i} \leftarrow Uniform(\{-2\epsilon, 2\epsilon\}^{h \times h})$
end

$$\begin{aligned}
\mathbb{E}|\langle \delta, v\rangle| &= \mathbb{E}|\sum_{k=r+1}^{r+h}\sum_{l=s+1}^{s+h}\sum_{i=1}^{c}\delta^i_{k,l}v^i_{k,l}| \\
&\overset{(i)}{=} \mathbb{E}_{(r,s)}\mathbb{E}_{\rho}|\sum_{k=r+1}^{r+h}\sum_{l=s+1}^{s+h}\sum_{i=1}^{c}\delta^i_{k,l}v^i_{k,l}| \\
&\overset{(ii)}{\geq} \frac{2\varepsilon}{\sqrt{2}}\mathbb{E}_{(r,s)}\|V_{(r,s)}\|_2 \\
&\overset{(iii)}{\geq} \sqrt{2}\varepsilon\|\mathbb{E}_{(r,s)}V_{(r,s)}\|_2 \\
&\geq \frac{\sqrt{2}\varepsilon h^2}{w^2}\|v\|_2,
\end{aligned}$$

where we define by $V_{(r,s)} = \{v^i_{k,l}\}_{k\in\{r+1,\dots,r+h\},l\in\{s+1,\dots,s+h\},i\in\{1,\dots,c\}}$ and (i) follows from the decomposition between the randomness of the Rademacher and the random window, (ii) follows from the Khintchine inequality and (iii) follows from Jensen inequality.

In addition we have for the variance:

$$\begin{aligned}
\mathbb{E}\|\delta\|_2^2 &= \mathbb{E}_{(r,s)}\sum_{k=r+1}^{r+h}\sum_{l=s+1}^{s+h}\sum_{i=1}^{c}\mathbb{E}_{\rho}(\delta^i_{k,l})^2 \\
&= \mathbb{E}_{(r,s)}\sum_{k=r+1}^{r+h}\sum_{l=s+1}^{s+h}\sum_{i=1}^{c}4\varepsilon^2 \\
&= 4c\varepsilon^2h^2.
\end{aligned}$$

Thus the assumptions in Eq. (3) hold for the sampling distribution P^{multiple}.

A.5 Why Updates of Equal Sign?

Proposition 2 underlines the importance of a large inner product $\mathbb{E}[|\langle \delta_t, \nabla g(x_t)\rangle|]$ in the direction of the gradients. This provides some intuition explaining why the update δ^{single}, where a single Rademacher is drawn for each window, is more efficient than the update δ^{multiple} where different Rademacher values are drawn. Following the observation that the gradients are often approximately piecewise constant [31], we consider, as a heuristic, a piecewise constant direction v for which we will show that

$$\mathbb{E}[|\langle \delta^{\text{single}}, v\rangle|] = \Theta(\|v\|_1) \;\text{ and }\; \mathbb{E}[|\langle \delta^{\text{multiple}}, v\rangle|] = \Theta(\|v\|_2).$$

Therefore the directions sampled by our proposal are more correlated with the gradient direction and help the algorithm to converge faster. This is also verified empirically in our experiments (see the ablation study in Sup. C).

Analysis. Let us consider the direction $v \in \mathbb{R}^{w \times w}$ composed of different blocks $\{V_{(r,s)}\}_{(r,s) \in \{0,\dots,w/h\}}$ of constant sign.

For this direction v we compare two different proposal P^{multiple} and P^{single} where we choose uniformly one random block (r, s) and we either assign a single Rademacher $\rho_{(r,s)}$ to the whole block (this is P^{single}) or we assign multiple Rademacher values $\{\rho_{(k,l)}\}_{k \in \{rh+1,\dots,(r+1)h\}, l \in \{sh+1,\dots,(s+1)h\}}$ (this is P^{multiple}). Using the Khintchine and Jensen inequalities similarly to Sec. A.4, we have

$$\begin{aligned}
\mathbb{E}|\langle \delta^{\text{multiple}}, v \rangle| &= \mathbb{E}| \sum_{k=rh+1}^{(r+1)h} \sum_{l=sh+1}^{(s+1)h} \delta_{k,l} v_{k,l}| \\
&\geq \frac{2\varepsilon}{\sqrt{2}} \mathbb{E}_{(r,s)} \|V_{(r,s)}\|_2 \\
&\geq \frac{\sqrt{2}\varepsilon h^2}{w^2} \|v\|_2 .
\end{aligned}$$

Moreover, we can show the following upper bound using the Khintchine inequality and the inequality between the l_1- and l_2-norms:

$$\begin{aligned}
\mathbb{E}|\langle \delta^{\text{multiple}}, v \rangle| &= \mathbb{E}| \sum_{k=rh+1}^{(r+1)h} \sum_{l=sh+1}^{(s+1)h} \delta_{k,l} v_{k,l}| \\
&\leq 2\varepsilon \mathbb{E}_{(r,s)} \|V_{(r,s)}\|_2 \\
&= \frac{2\varepsilon h^2}{w^2} \sum_{r=1}^{w/h} \sum_{s=1}^{w/h} \|V_{(r,s)}\|_2 \\
&\leq \frac{2\varepsilon h}{w} \|v\|_2
\end{aligned}$$

Thus, $\mathbb{E}[|\langle \delta^{\text{multiple}}, v \rangle|] = \Theta(\|v\|_2)$.

For the update δ^{single} we obtain

$$\begin{aligned}
\mathbb{E}|\langle \delta^{\text{single}}, v \rangle| &= \mathbb{E}| \sum_{k=rh+1}^{(r+1)h} \sum_{l=sh+1}^{(s+1)h} \delta_{r,s} v_{k,l}| \\
&= \mathbb{E}|\delta_{r,s} \sum_{k=rh+1}^{(r+1)h} \sum_{l=sh+1}^{(s+1)h} v_{k,l}| \\
&\overset{(i)}{=} 2\varepsilon \mathbb{E}_{(r,s)} \|V_{(r,s)}\|_1 \\
&= \frac{2\varepsilon h^2}{w^2} \|v\|_1
\end{aligned}$$

where (i) follows from the fact the $V_{(r,s)}$ has a constant sign. We recover then the l_1-norm of the direction v, i.e. we conclude that $\mathbb{E}[|\langle \delta^{\text{single}}, v \rangle|] = \Theta(\|v\|_1)$.

This implies that for an approximately constant block $\mathbb{E}|\langle \delta^{\text{single}}, v \rangle|$ will be larger than $\mathbb{E}|\langle \delta^{\text{multiple}}, v \rangle|$. For example, in the extreme case of constant binary

block $|V_{(r,s)}| = 11^\top$, we have

$$\mathbb{E}|\langle \delta^{\text{single}}, v\rangle| = 2\varepsilon h^2 >> \mathbb{E}|\langle \delta^{\text{multiple}}, v\rangle| \asymp 2\varepsilon h.$$

A.6 Proof of Proposition 3

Let $x \in \mathcal{F}$, and $N(x)$ the number of elements of $\mathcal{G}$ that x contains. Let initialize x as a square of size $s \times s$, so that $N(x) = 1$. We then add iteratively the remaining $k - s^2$ unitary squares to x so to maximize $N(x)$.
In order to get $N(x) = 2$ it is necessary to increase x to have size $s \times (s + 1)$. At this point, again to get $N(x) = 3$ we need to add s squares to one side of x. However, if we choose to glue them so to form a rectangle $s \times (s + 2)$, then $N(x) = 3$ and once more we need other s squares to increase N, which means overall $s^2 + 3s$ to achieve $N(x) = 4$. If instead we glue s squares along the longer side, with only one additional unitary square we get $N(x) = 4$ using $s^2 + 2s + 1 < s^2 + 3s$ unitary squares (as $s \geq 2$), with $x = (s + 1) \times (s + 1)$.
Then, if the current shape of x is $a \times b$ with $a \geq b$, the optimal option is adding a unitary squares to have shape $a \times (b+1)$, increasing the count N of $a - s + 1$. This strategy can be repeated until the budget of k unitary squares is reached.
Finally, since we start from the shape $s \times s$, then at each stage $b - a \in \{0, 1\}$, which means that the final a will be $\left\lfloor \sqrt{k} \right\rfloor$. A rectangle $a \times b$ in $\mathcal{F}$ contains $(a - s + 1)(b - s + 1)$ elements of $\mathcal{G}$. The remaining $k - ab$ squares can be glued along the longer side, contributing to $N(x)$ with $(k - ab - s + 1)^+$.

B Experimental Details

In this section, we list the main hyperparameters and various implementation details for the experiments done in the main experiments (Sec. 5).

B.1 Experiments on ImageNet

For the untargeted Square Attack on the ImageNet models, we used $p = 0.05$ and $p = 0.1$ for the l_∞- and l_2- versions respectively. For Bandits, we used their code with their suggested hyperparameters (specified in the configuration files) for both l_∞ and l_2. For SignHunter, we used directly their code which does not have any hyperparameters (assuming that the finite difference probe δ is set to ϵ). For SimBA-DCT, we used the default parameters of the original code apart from the following, which are the suggested ones for each model: for ResNet-50 and VGG-16-BN "freq_dims" = 28, "order" = "strided" and "stride" = 7, for Inception v3 "freq_dims" = 38, "order" = "strided" and "stride" = 9. Notice that SimBA tries to minimize the l_2-norm of the perturbations but it does not have a bound on the size of the changes. Then we consider it successful when the adversarial examples produced have norm smaller than the fixed threshold ϵ. The results for all other methods were taken directly from the corresponding papers.

Evaluation of Bandits. The code of Bandits [31] does not have image standardization at the stage where the set of correctly points is determined (see `https://github.com/MadryLab/blackbox-bandits/issues/3`). As a result, the attack is run only on the set of points correctly classified by the network *without standardization*, although the network was trained on standardized images. We fix this bug, and report the results in Table 2 based on the fixed version of their code. We note that the largest difference of our evaluation compared to the l_∞ results reported in Appendix E of [31] is obtained for the VGG-16-BN network: we get 2.0% failure rate while they reported 8.4% in their paper. Also, we note that the query count for Inception v3 we obtain is also better than reported in [31]: 957 instead of 1117 with a slightly better failure rate. Our l_2 results also differ – we obtain a significantly lower failure rate (9.8%, 6.8%, 10.2% instead of 15.5%, 9.7%, 17.2% for the Inception v3, ResNet-50, VGG-16-BN networks respectively) with improved average number of queries (1486, 939, 511 instead of 1858, 993, 594).

B.2 Square Attack Can be More Accurate than White-box Attacks

For the l_∞-Square Attack, we used $p = 0.3$ for all models on MNIST and CIFAR-10. For Bandits on MNIST and CIFAR-10 adversarially trained models we used "exploration" = 0.1, "tile size" = 16, "gradient iters" = 1 following [49].

For the comparison of l_2-attacks on the l_∞-adversarially trained model of [37] we used the Square Attack with the usual parameter $p = 0.1$. For Bandits we used the parameters "exploration" = 0.01, "tile size" = 28, "gradient iters" = 1, after running a grid search over the three of them (all the other parameters are kept as set in the original code). For SimBA we used the "pixel attack" with parameters "order" = "rand", "freq_dims" = 28, step size of 0.50, after a grid search on all the parameters.

C Ablation Study

Here we discuss in more detail the ablation study which justifies the algorithmic choices made for our l_∞- and l_2-attacks. Additionally, we discuss the robustness of the attack to the hyperparameter p, i.e. the initial fraction of pixels changed by the attack (see Fig. 5). We perform all these experiments on ImageNet with a standardly trained ResNet-50 model from the PyTorch repository.

C.1 l_∞-Square Attack

Sensitivity to the hyperparameter p. First of all, we note that for *all* values of p we achieve 0.0% failure rate. Moreover, we achieve state-of-the-art query efficiency with *all* considered values of p (from 0.0125 to 0.4), i.e. we have the average number of queries below 140, and the median below 20 queries. Therefore, we conclude that the attack is robust to a wide range of p, which is an important property of a black-box attack – since the target model is unknown,

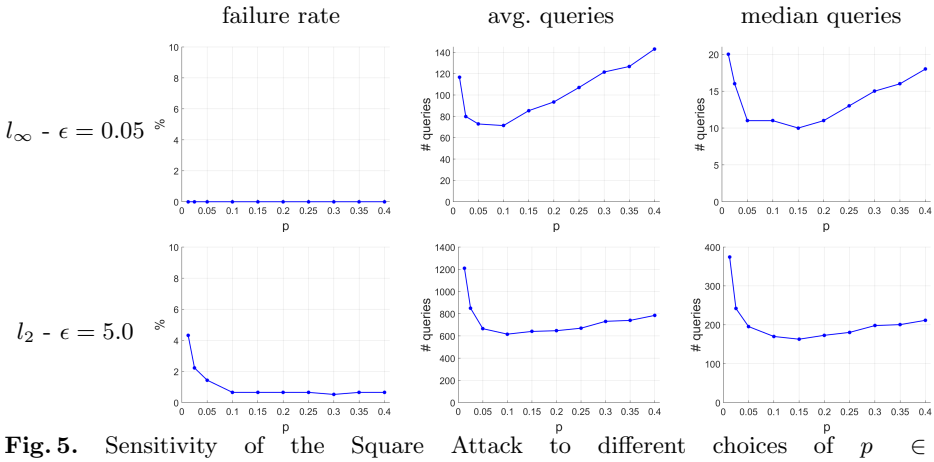

Fig. 5. Sensitivity of the Square Attack to different choices of $p \in \{0.0125, 0.025, 0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4\}$, i.e. the initial fraction of pixels changed by the attack, on ImageNet for a ResNet-50 model

and one aims at minimizing the number of queries needed to fool the model, doing even an approximate grid search over p is prohibitively expensive.

Algorithmic choices. In addition to the results presented in Sec. 4.3, we show in Table 7 the results of a few more variants of the Square Attack. We recall that "# random signs" indicates how many different signs we sample to build the updates, with c being the number of color channels and h the current size of the square-shaped updates. Specifically, we test the performance of using a single random sign for all the elements for the update, "square-1", which turns out to be comparable to "square-$c \cdot h^2$", i.e. every component of the update has sign independently sampled, but worse than keeping the sign constant within each color channel ("square-c").

In order to implement update shape "rectangle", on every iteration and for every image we sample $\alpha, \beta \sim \text{Exp}(1)$ and take a rectangle with sides $\alpha \cdot s$ and $\beta \cdot s$, so that in expectation its area is equal to s^2, i.e. to the area of the original square. This update scheme performs significantly better than changing a random subset of pixels (93 vs 339 queries on average), but worse than changing squares (73 queries on average) as discussed in Sec. 4.2.

Finally, we show the results with two more initialization schemes: horizontal stripes (instead of vertical), as well as initialization with randomly placed squares. While both solutions lead to the state-of-the-art query efficiency (83 and 90 queries on average) compared to the literature, they achieve worse results than the vertical stripes we choose for our Square Attack.

C.2 l_2-Square Attack

Sensitivity to the hyperparameter p. We observe that the l_2-Square Attack is robust to different choices in the range between 0.05 and 0.4 showing approximately the same failure rate and query efficiency for all values of p in

Table 7. An ablation study for the performance of the l_∞- and l_2-Square Attack under various algorithmic choices of the attack. The metrics are calculated on 1,000 ImageNet images for a ResNet-50 model. The last row represents our recommended setting. For all experiments we used the best performing p (0.05 for l_∞ and 0.1 for l_2)

l_∞ ablation study

Update shape	# random signs	Initialization	Failure rate	Avg. queries	Median queries
random	$c \cdot h^2$	vert. stripes	0.0%	401	48
random	$c \cdot h^2$	uniform rand.	0.0%	393	132
random	c	vert. stripes	0.0%	339	53
square	$c \cdot h^2$	vert. stripes	0.0%	153	15
square	1	vert. stripes	0.0%	129	18
rectangle	c	vert. stripes	0.0%	93	16
square	c	uniform rand.	0.0%	91	26
square	c	rand. squares	0.0%	90	20
square	c	horiz. stripes	0.0%	83	18
square	c	vert. stripes	0.0%	73	11

l_2 ablation study

Update	Initialization	Failure rate	Avg. queries	Median queries
η^{rand}	η^{rand}-grid	3.3%	1050	324
η^{single}	η^{single}-grid	0.7%	650	171
η	gaussian	0.4%	696	189
η	uniform	0.8%	660	187
η	vert. stripes	0.8%	655	186
η	η-grid	0.7%	616	170

this range, while its performance degrades slightly for very small initial squares $p \in \{0.0125, 0.025\}$.

Algorithmic choices. We analyze in Table 7 the sensitivity of the l_2-attack to different choices of the shape of the update and initialization.

In particular, we test an update with only one "center" instead of two, namely $\eta^{\text{single}} = \eta^{h,h}$ (following the notation of Eq. 2) and one, η^{rand}, where the step 7 in Algorithm 3 is $\rho \leftarrow Uniform(\{-1, 1\}^{h \times h})$ instead of $\rho \leftarrow Uniform(\{-1, 1\})$, which means that each element of η is multiplied randomly by either -1 or 1 independently (instead of all elements multiplied by the same value). We can see that using different random signs in the update and initialization (η^{rand}) significantly (1.5× factor) degrades the results for the l_2-attack, which is similar to the observation made for the l_∞-attack.

Alternatively to the grid described in Sec. 3.4, we consider as starting perturbation i) a random point sampled according to $Uniform(\{-\epsilon/\sqrt{d}, \epsilon/\sqrt{d}\}^{w \times w \times c})$, that is on the corners of the largest l_∞-ball contained in the l_2-ball of radius ϵ (uniform initialization), ii) a random position on the l_2-ball of radius ϵ (Gaussian initialization) or iii) vertical stripes similarly to what done for the l_∞-Square Attack, but with magnitude $\epsilon/\sqrt{d}$ to fulfill the constraints on the l_2-norm of the

Table 8. Mean and standard deviation of the main performance metrics of the Square Attack across 10 different runs with different random seeds

ImageNet, ResNet-50

Norm	ϵ	Failure rate	Avg. queries	Median queries
l_∞	0.05	$0.0\% \pm 0.0\%$	72 ± 2	11 ± 1
l_2	5	$0.6\% \pm 0.1\%$	638 ± 12	163 ± 8

MNIST, adversarially trained LeNet from [37]

Norm	ϵ	Robust accuracy	Avg. queries	Median queries
l_∞	0.3	$87.0\% \pm 0.1\%$	299 ± 47	52 ± 7
l_2	2	$16.0\% \pm 1.4\%$	1454 ± 71	742 ± 78

perturbation. We note that different initialization schemes do not have a large influence on the results of our l_2-attack, unlike for the l_∞-attack.

D Stability of the Attack under Different Random Seeds

Here we study the stability of the Square Attack over the randomness in the algorithm, i.e. in the initialization, in the choice of the locations of square-shaped regions, and in the choice of the values in the updates δ. We repeat 10 times experiments similar to the ones reported in Sec. 5.1 and Sec. 5.2 with different random seeds for our attack, and report all the metrics with standard deviations in Table 8. On ImageNet, we evaluate the *failure rate* (over initially correctly classified points) and query efficiency on 1,000 images using ResNet-50. On MNIST, we evaluate the *robust accuracy* (i.e. the failure rate over all points) and query efficiency on 1,000 images using the l_∞-adversarially trained LeNet from [37]. Note that unlike in Sec. 5.2 in both cases we use a single restart for the attack on MNIST, and we compute the statistics on 1,000 points instead of 10,000, thus the final results will differ.

On the ImageNet model, all these metrics are very concentrated for both the l_∞- and the l_2-norms. Moreover, we note that the standard deviations are much smaller than the gap between the Square Attack and the competing methods reported in Table 2. Thus we conclude that the results of the attack are stable under different random seeds.

On the adversarially trained MNIST model from [37], the robust accuracy is very concentrated showing only 0.1% and 1.4% standard deviations for the l_∞- and the l_2-norms respectively. Importantly, this is much less than the gaps to the nearest competitors reported in Tables 4 and 5. We also show query efficiency for this model, although for models with non-trivial robustness it is more important to achieve lower robust accuracy, and query efficiency on successful adversarial examples is secondary. We note that the standard deviation of the mean and median number of queries is higher than for ImageNet, particularly for the l_∞-ball of radius $\epsilon = 0.3$ where the robust accuracy is much higher than for the l_2-ball of radius $\epsilon = 2$. This is possibly due to the fact that attacking more

robust models (within a certain threat model) is a more challenging task than, e.g., attacking standardly trained classifiers, as those used on ImageNet, which means that a favorable random initialization or perturbation updates can have more influence on the query efficiency.

E Additional Experimental Results

This section contains results on targeted attacks, and also additional results on untargeted attacks that complement the ImageNet results from Table 2. Moreover, we show that the Square Attack is useful for evaluating the robustness of newly proposed defenses (see Sec. E.6).

E.1 Targeted Attacks

While in Sec. 5 we considered only untargeted attacks, here we report the results of the different attacks in the *targeted* scenario.

Targeted Square Attack. In order to adapt our scheme to targeted attacks, where one first choose a target class t and then tries to get the model f to classify a point x as t, we need to modify the loss function L which is minimized (see Eq. (1)). For the untargeted attacks we used the margin-based loss $L(f(x), y) = f_y(x) - \max_{k \neq y} f_k(x)$, with y the correct class of x. This loss could be straightforwardly adapted to the targeted case as $L(f(x), t) = -f_t(x) + \max_{k \neq t} f_k(x)$. However, in practice we observed that this loss leads to suboptimal query efficiency. We hypothesize that the drawback of the margin-based loss in this setting is that the maximum over $k \neq t$ is realized by different k at different iterations, and then the changes applied to the image tend to cancel each other. We observed this effect particularly on ImageNet which has a very high number of classes.

Instead, we use here as objective function the cross-entropy loss on the target class, defined as

$$L(f(x), t) = -f_t(x) + \log \left(\sum_{i=1}^{K} e^{f_i(x)} \right). \tag{5}$$

Minimizing L is then equivalent to maximizing the confidence of the classifier in the target class. Notice that in Eq. (5) the scores of all the classes are involved, so that it increases the *relative* weight of the target class respect to the others, making the targeted attacks more effective.

Experiments. We present the results for targeted attacks on ImageNet for Inception-v3 model in Table 9. We calculate the statistics on 1,000 images (the target class is randomly picked for each image) with query limit of 100,000 for l_∞ and on 200 points and with query limit 60,000 for l_2, as this one is more expensive computationally because of the lower success rate, with the same norm bounds ϵ used in the untargeted case. We use the Square Attack with $p = 0.01$ for l_∞ and $p = 0.02$ for l_2. The results for the competing methods for l_∞ are taken from [39],

Table 9. Results of **targeted** attacks on ImageNet for Inception-v3 model using 100k query limit for l_∞, 60k for l_2. The results for the competing methods for l_∞ are taken from [39], except SignHunter [2] which we evaluated using their code

Norm	Attack	Failure rate	Avg. queries	Median queries
l_∞	Bandits [31]	7.5%	25341	18053
	SignHunter [2]	1.1%	8814	5481
	Parsimonious [49]	**0.0%**	7184	5116
	DFO_c – Diag. CMA [39]	6.0%	6768	3797
	DFO_c – CMA [39]	**0.0%**	6662	4692
	Square Attack	**0.0%**	**4584**	**2859**
l_2	Bandits [31]	**24.5%**	20489	17122
	SimBA-DCT [28]	25.5%	30576	30180
	Square Attack	33.5%	**19794**	**15946**

except SignHunter [2] which was not evaluated in the targeted setting before, thus we performed the evaluation using their code on 100 test points using the cross-entropy as the loss function. For l_2, we use Bandits [31] with the standard parameters used in the untargeted scenario, while we ran a grid search over the step size of SimBA (we set it to 0.03) and keep the other hyperparameters as suggested for the Inception-v3 model.

The targeted l_∞-Square Attack achieves 100% success rate and requires 1.5 times fewer queries on average than the nearest competitor [39], showing that even in the *targeted* scenario our simple scheme outperforms the state-of-the-art methods. On the other hand, our l_2-attack suffers from worse higher failure rate than the competitors, but achieves lower average and median number of queries (although on a different number of successful points).

E.2 Success Rate on ImageNet for Different Number of Queries

In this section, we provide a more detailed comparison to the competitors from Table 2 under different query budgets and more comments about the low query regime experiment in Fig. 4. We show in Fig. 6 the behaviour of the success rate for each attack depending on the number queries. The success rates of the attacks from [39] (DFO_c–CMA–50 and DFO_d–Diag. CMA–30) and [50] (BayesAttack) for different number of queries were obtained via personal communication directly from the authors, and were calculated on 500 and 10,000 randomly sampled points, respectively. For the other attacks, as mentioned above, the success rate is calculated on 1,000 randomly sampled points.

l_∞-results. First, we observe that the Square Attack outperforms all other methods in the standard regime with 10,000 queries. The gap in the success rate gets larger in the range of 100-1000 queries for the more challenging Inception-v3 model, where we observe over 10% improvement in the success rate over all other methods including SignHunter. Our method also outperforms the BayesAttack in the low query regime, i.e. less than 200 queries, by approximately 20% on every model. We note that DFO_d–Diag. CMA–30 method is also quite effective

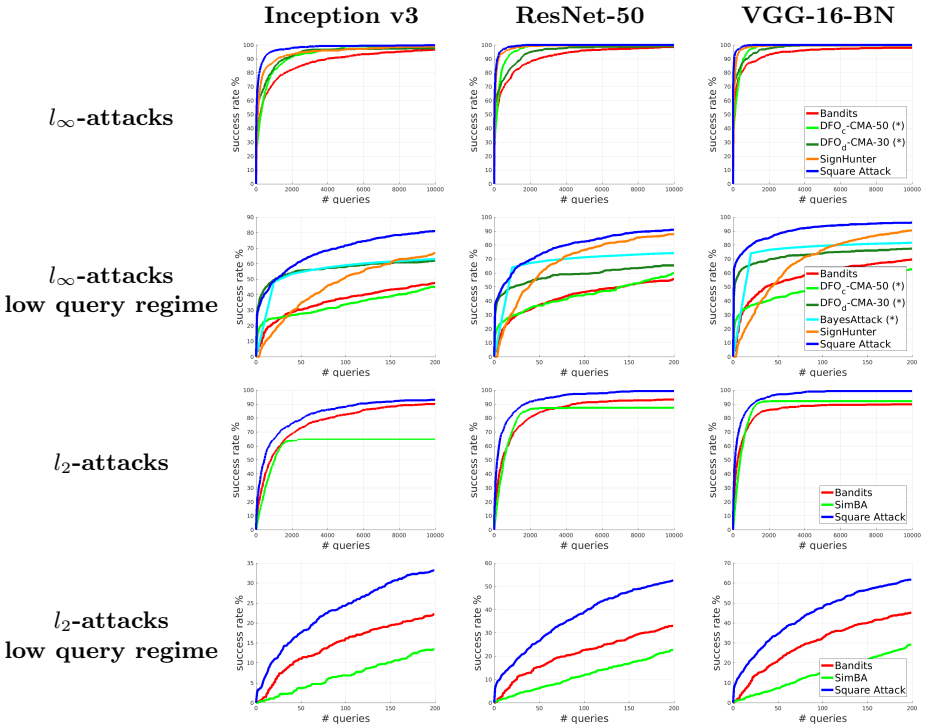

Fig. 6. Success rate vs number of queries for different attacks on ImageNet on three standardly trained models. The low query regime corresponds to up to 200 queries, while the standard regime corresponds to 10,000 queries. * denotes the results obtained via personal communication with the authors and evaluated on 500 and 10,000 randomly sampled points for DFO [39] and BayesAttack [50] methods, respectively

in the low query regime showing results close to BayesAttack. However, it is also outperformed by our Square Attack.

l_2-results. First, since the l_2-version of SignHunter [2] is not competitive to Bandits on ImageNet (see Fig. 2 in [2]), we do not compare to them here. The l_2-Square Attack outperforms both Bandits and SimBA, and the gap is particularly large in the low query regime. We note that the success rate of SimBA plateaus after some iteration. This happens due to the fact that their algorithm only adds orthogonal updates to the perturbation, and does not have any way to correct the greedy decisions made earlier. Thus, there is no progress anymore after the norm of the perturbation reaches the $\epsilon = 5$ (note that we used for SimBA the same parameters of the comparison between SimBA and Bandits in [28]). Contrary to this, both Bandits and our attack constantly keep improving the success rate, although with a different speed.

Table 10. Results of untargeted l_∞-perturbations produced by the Square Attack on architectures with dilated convolutions

ϵ_∞	Model	Failure rate	Avg. queries	Median queries
0.05	DRN-A-50	0.0%	86	12
	DRN-C-42	0.0%	57	7
	DRN-D-38	0.0%	48	6

E.3 Performance on Architectures with Dilated Convolutions

In Sec. 4.2, we provided justifications for square-shaped updates for *convolutional* networks. Thus, a reasonable question is whether the Square Attack still works equally well on less standard convolutional networks such as, for example, networks with dilated convolutions. For this purpose, we evaluate three different architectures introduced in [59] that involve *dilated* convolutions: DRN-A-50, DRN-C-42 and DRN-D-38. We use $\epsilon_\infty = 0.05$ as in the main ImageNet experiments from Table 2. We present the results in Table 10 and observe that for all the three model the Square Attack achieves 100% success rate and both average and median number of queries stay comparable to that of VGG or ResNet-50 from Table 2. Thus, this experiment suggests that our attack can be applied not only to standard convolutional networks, but also to more recent neural network architectures.

E.4 Imperceptible Adversarial Examples with the Square Attack

Adversarial examples in general need not be imperceptible, for example adversarial patches [10,34] are clearly visible, and yet can be used to attack machine learning systems deployed in-the-wild. However, if imperceptibility is the goal, it can be easily ensured by adjusting the size of the allowed perturbations. In the main ImageNet experiments in Table 2 we used $\epsilon_\infty = 0.05 = 12.75/255$ since this is standard in the literature [2,31,39,49], and for which all the considered attacks produce visible perturbations. Below we additionally provide results on the more than $3\times$ smaller threshold $\epsilon_\infty = 4/255$ which leads to imperceptible perturbations (see Fig. 7). Our attack still achieves almost perfect success rate requiring only a limited number of queries as shown in Table 11. Thus, one can also generate imperceptibile adversarial examples with the Square Attack simply by adjusting the perturbation size.

Table 11. Results of untargeted imperceptible l_∞-perturbations produced by the Square Attack on standard architectures

ϵ_∞	Model	Failure rate	Avg. queries	Median queries
4/255	VGG	0.5%	424	115
	ResNet-50	0.3%	652	213
	Inception v3	5.4%	1013	391

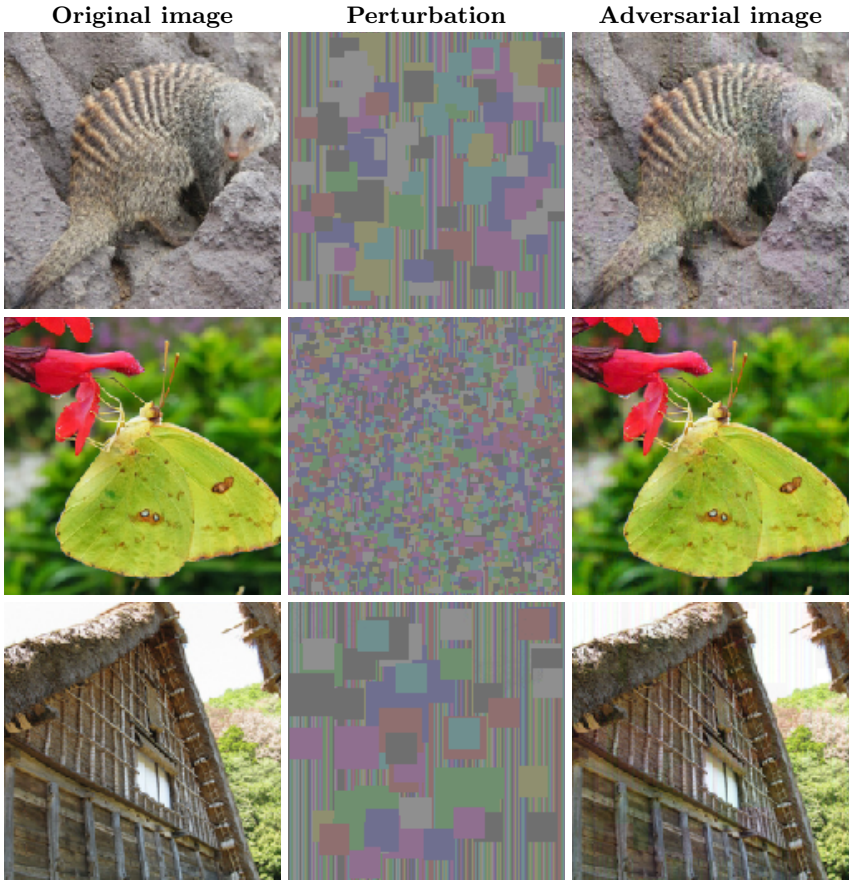

Fig. 7. Visualization of the *imperceptible* adversarial examples found by the l_∞ Square Attack on ImageNet using ResNet-50 for $\epsilon_\infty = 4/255$. All the original images were correctly classified while the adversarial images are misclassified by the model. The perturbations are amplified for the visualization purpose

E.5 Analysis of Adversarial Examples that Require More Queries

Here we provide more visualizations of adversarial perturbations generated by the untargeted Square Attack for $\epsilon = 0.05$ on ImageNet. We analyze here the inputs that require more queries to be misclassified. We present the results in Fig. 8 where we plot adversarial examples after 10, 100 and 500 iterations of our attack. First, we note that a misclassification is achieved when the margin loss becomes negative. We can observe that the loss decreases gradually over iterations, and a single update only rarely leads to a significant decrease of the loss. As the attack progresses over iterations, the size of the squares is reduced according to our piecewise-constant schedule leading to more refined perturbations since the algorithm accumulates a larger number of square-shaped updates.

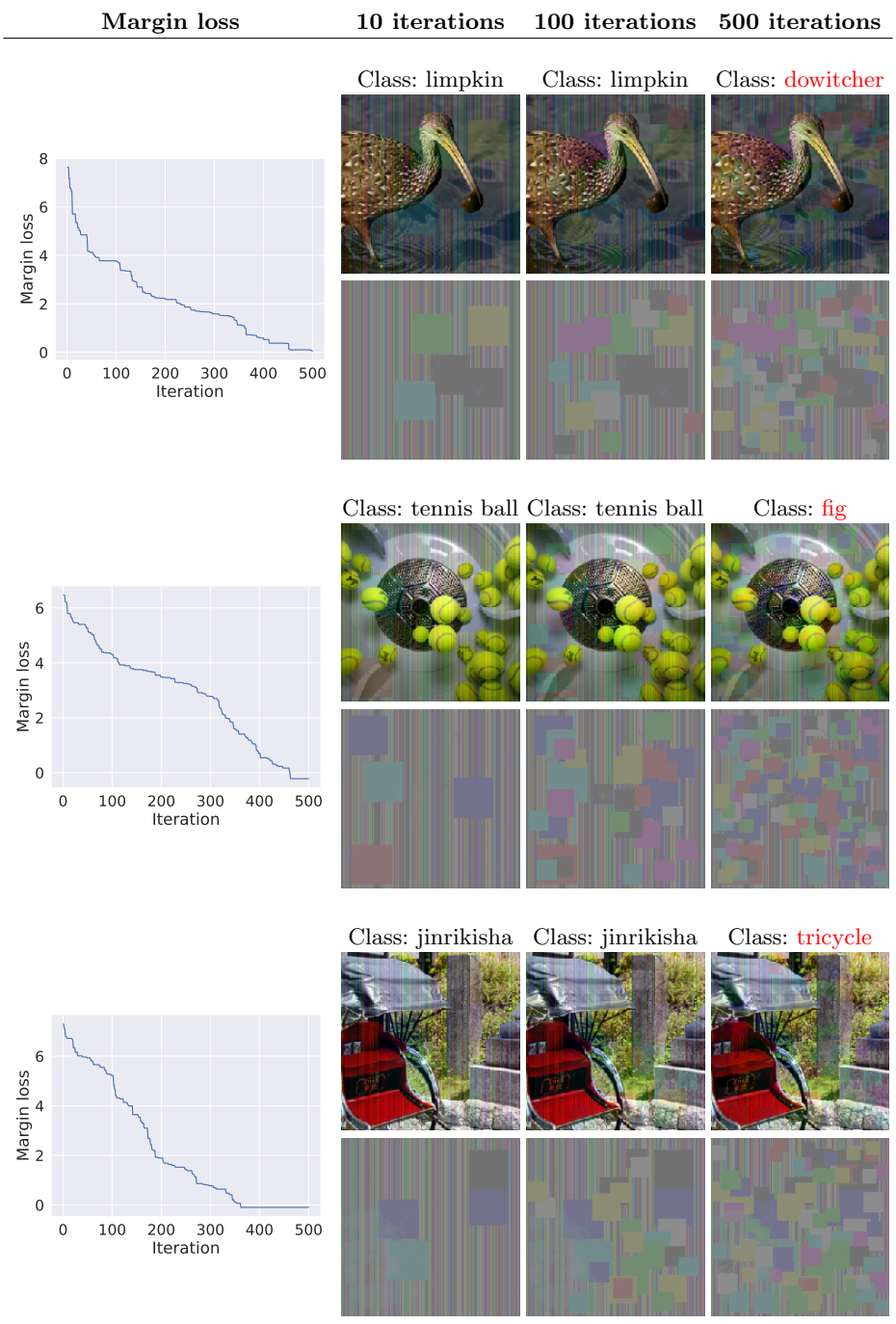

Fig. 8. Visualization of adversarial examples for which the untargeted l_∞ Square Attack requires more queries. We visualize adversarial examples and perturbations after 10, 100 and 500 iterations of the attack. The experiment is done on ImageNet using ResNet-50 for $\epsilon_\infty = 0.05$. Note that a misclassification is achieved when the margin loss becomes negative

E.6 Breaking the Post-averaging Defense

We investigate whether the l_∞-robustness claims of [36] hold (as reported in `https://www.robust-ml.org/preprints/`). Their defense method is a randomized averaging method similar in spirit to [18]. The difference is that [36] sample from the surfaces of several d-dimensional spheres instead of the Gaussian distribution, and they do not derive any robustness certificates, but rather measure robustness by the PGD attack. We use the hyperparameters specified in their code (K=15, R=6 on CIFAR-10 and K=15, R=30 on ImageNet). We show in Table 12 that the proposed defense can be broken by the l_∞-Square Attack, which is able to reduce the robust accuracy suggested by PGD from 88.4% to 15.8% on CIFAR-10 and from 76.1% to 0.4% on ImageNet (we set $p = 0.3$ for our attack). This again highlights that straightforward application of gradient-based white-box attacks may lead to inaccurate robustness estimation, and usage of the Square Attack can prevent false robustness claims.

Table 12. l_∞-robustness of the post-averaging randomized defense [36]. The Square Attack shows that these models are not robust

ϵ_∞	**Dataset**	**Robust accuracy**		
		Clean	PGD	**Square**
8/255	CIFAR-10	92.6%	88.4%	**15.8%**
	ImageNet	77.3%	76.1%	**0.4%**