Optimization and Engineering (2022) 23:1319–1346
https://doi.org/10.1007/s11081-021-09652-w

REVIEW ARTICLE

An empirical study of derivative-free-optimization algorithms for targeted black-box attacks in deep neural networks

Giuseppe Ughi[1] · Vinayak Abrol[1] · Jared Tanner[1]

Received: 23 September 2020 / Revised: 7 June 2021 / Accepted: 7 June 2021 /
Published online: 30 June 2021

Abstract

We perform a comprehensive study on the performance of derivative free optimization (DFO) algorithms for the generation of targeted black-box adversarial attacks on Deep Neural Network (DNN) classifiers assuming the perturbation energy is bounded by an ℓ_∞ constraint and the number of queries to the network is limited. This paper considers four pre-existing state-of-the-art DFO-based algorithms along with a further developed algorithm built on BOBYQA, a model-based DFO method. We compare these algorithms in a variety of settings according to the fraction of images that they successfully misclassify given a maximum number of queries to the DNN. The experiments disclose how the likelihood of finding an adversarial example depends on both the algorithm used and the setting of the attack; algorithms limiting the search of adversarial example to the vertices of the ℓ^∞ constraint work particularly well without structural defenses, while the presented BOBYQA based algorithm works better for especially small perturbation energies. This variance in performance highlights the importance of new algorithms being compared to the state-of-the-art in a variety of settings, and the effectiveness of adversarial defenses being tested using as wide a range of algorithms as possible.

Keywords Derivative free optimization · Deep learning · Black-box attacks

This publication is based on work supported by the EPSRC Centre for Doctoral Training in Industrially Focused Mathematical Modelling (EP/L015803/1) in collaboration with New Rock Capital Management and by the Alan Turing Institute under the EPSRC grant EP/N510129/1.

✉ Giuseppe Ughi
ughi@maths.ox.ac.uk

Vinayak Abrol
abrol@maths.ox.ac.uk

Jared Tanner
tanner@maths.ox.ac.uk

[1] Mathematical Institute, University of Oxford, Andrew Wiles Building, Radcliffe Observatory Quarter, Oxford OX2 6GG, United Kingdom

1 Introduction

Deep Neural Networks (DNNs) achieve state-of-the-art performance on a growing number of applications such as acoustic modelling (Hinton et al. 2012), image classification (He et al. 2015), and fake news detection (Monti et al. 2019) to name but a few. Alongside their growing application, there is a literature on the robustness of deep networks which shows that it is often possible to subtly perturb the input image of a DNN in order to degrade its performance; these perturbations are referred to as adversarial examples (Goodfellow et al. 2015; Szegedy et al. 2014). For example, see (Dalvi et al. 2004; Eykholt et al. 2018; Kurakin et al. 2017; Sitawarin et al. 2018; Yuan et al. 2019) where road signals are perturbed so as to be wrongly interpreted by self driving cars that analyze images of them with DNNs. Methods to generate these adversarial examples are classified according to two main criteria (Yuan et al. 2019):

Adversarial Specificity establishes what the aim of the adversary is. In *non-targeted* attacks, the method perturbs the image in such a way that it is misclassified into any category other than the original one. While in *targeted* settings, the adversary specifies a category into which an image should be misclassified.

Adversary's Knowledge defines the amount of information available to the adversary. In *White-box* settings the adversary has complete knowledge of the network architecture and weights, while in the *Black-box* setting the adversary is only able to obtain the pre-classification output vector. The White-box setting allows for the use of gradients of a misclassification objective to efficiently compute the adversarial example (Carlini and Wagner 2017; Chen et al. 2018; Goodfellow et al. 2015), while the same optimization formulation of the Black-box setting requires use of a derivative free approach (Alzantot et al. 2019; Chen et al. 2017; Ilyas et al. 2018; Narodytska and Kasiviswanathan 2017).

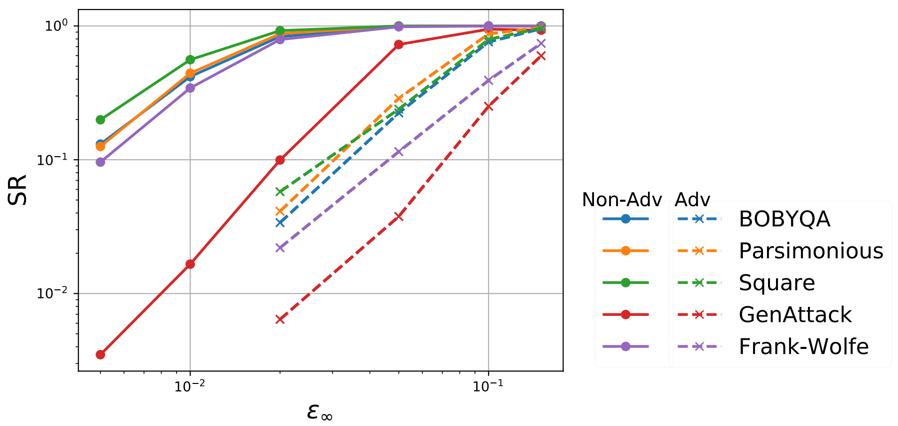
In this work we consider the targeted black-box setting. In particular we follow Chen et al. (2017) where:

- the *perturbation*, which causes the network to change the classification, is bounded in magnitude by a specified ℓ^∞-norm, ε_∞, i.e. each pixel in the image cannot be perturbed by more than ε_∞;
- the *number of queries* to the DNN needed to generate a targeted adversarial example should be as small as possible.

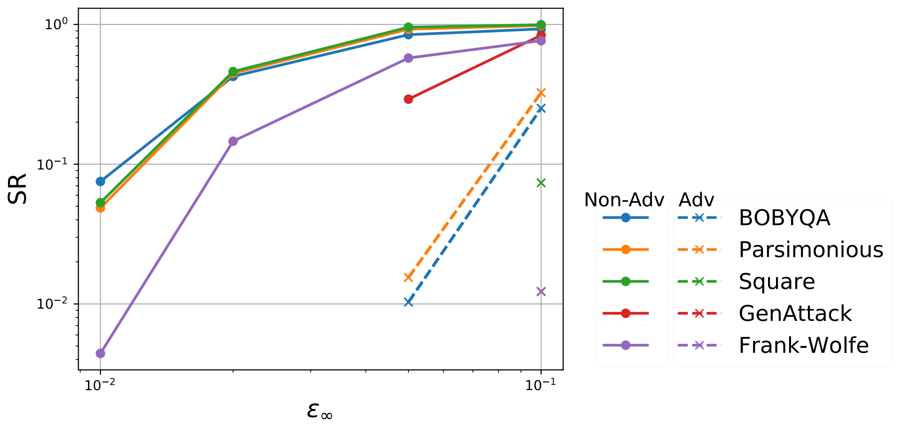
The Zeroth-Order-optimization (ZOO) algorithm proposed in Chen et al. (2017) describes a Derivative Free optimization (DFO) method for computing adversarial examples in the black-box setting using a coordinate descent optimization method. At the time this was a substantial departure from previous black-box algorithms which trained a proxy DNN and then employ gradient based white-box attacks on the proxy network (Papernot et al. 2017; Tu et al. 2018). It was demonstrated in Chen et al. (2017) that these algorithms are especially effective when numerous adversarial examples are computed, but become less efficient when an individual adversarial examples is considered. Following the introduction of ZOO, there have been numerous improvements using other model-free DFO based approaches, see for example (Al-Dujaili and O'Reilly 2020; Alzantot et al. 2019; Andriushchenko et al. 2020; Chen et al. 2020; Ilyas et al. 2018, 2019; Moon et al. 2019). Many of these algorithms were developed in parallel, and so have not yet been bench-marked in a consistent setting, e.g. on the same network.

In this article, we present two frameworks for comparative evaluation of the existing algorithms that claim to have the fewest number of DNN queries to generate a successful attack. These are: GenAttack (Alzantot et al. 2019) which is based on a genetic direct-search method; Parsimonious algorithm (Moon et al. 2019), based on a combinatorial direct-search method on the vertices of the perturbation domain; the Square algorithm (Andriushchenko et al. 2020), based on a randomized direct-search method on the vertices of the perturbation domain; the Frank-Wolfe algorithm (Chen et al. 2020) based on a momentum mechanism that approximates the gradient via finite differences; and BOBYQA (Ughi et al. 2019), which explicitly develops models to approximate the loss function and then minimizes the model over a trust region using techniques from continuous optimization. The aforementioned list of algorithms covers the leading classes of DFO algorithms for limited function evaluations, see e.g., (Conn et al. 2009; Larson et al. 2019) for recent reviews of DFO methods. The two frameworks are structured as follows:

1. In the first setting we consider attacks on DNNs trained on CIFAR10 and ImageNet datasets, with or without the adversarial defense by MadryLab (Engstrom et al. 2019); this is the canonical setup for the comparison of black-box attacks that was considered in previous literature. We illustrate in Fig. 1 a measure of how the performance of the considered algorithms compare, while further refined measures of comparison are included in Sect. 4. We observe that the algorithms that limit the optimization domain to the ℓ^∞ perturbation boundary, i.e. the Parsimonious and Square algorithms, are consistently the most effective. In particular, the Square algorithm achieves the highest Success Ratio (SR) with a fixed maximum number of queries, except for when the DNNs have been adversarially trained, and the Parsimonious algorithm achieves the highest SR when a network is trained with the MadryLab defense. However, these results are relative to the current state-of-the-art defense in a field which is in continuous development (Dhillon et al. 2018; Wang et al. 2019) and newly proposed methods usually have a varying effect on the different attacking algorithms; for example the MadryLab



(a) Cifar10



(b) ImageNet

Fig. 1 The success rate (SR) of targeted attacks as a function of the perturbation's allowed ℓ^∞ magnitude for algorithms: GenAttack (Alzantot et al. 2019), Parsimonious (Moon et al. 2019), Square (Andriushchenko et al. 2020), Frank-Wolfe (Chen et al. 2020), and the BOBYQA based algorithm further developed here. Specifically for a ResNet50 network trained either on the CIFAR10 (a) or the ImageNet (b) dataset with (Adv) and without (Non-Adv) the defense by MadryLab Engstrom et al. (2019). An attack is considered successful if the method found the targeted adversarial example with less than 3'000 or 15'000 queries to the network trained on CIFAR and ImageNet dataset, respectively. Results for the case SR=0, i.e. when no perturbations were successful, are excluded from the plot

defense (Engstrom et al. 2019) that we consider is most effective on Square algorithm in the ImageNet case.

2. In the second framework, the algorithms are allowed to perturb only a fraction of the pixels in the input; this is especially inspired by the structural defenses that transform the input in the wavelet space (Guo et al. 2018). This framework allows us to understand the sensitivity of different algorithms to choices such as initialization, experimental protocol, dataset, and adversarial training. Our results

demonstrate that the Parsimonious, Square, and BOBYQA based algorithms alternatively perform the best for different maximum perturbation energies.

The results in this paper show that the most likely algorithm to find an adversarial example varies according to the considered setting; the type of dataset, the defense, and the perturbation energy bound have a varying impact on the different algorithms. As a consequence of these experiments, new algorithms should be compared to the state-of-the-art in a variety of settings as done here, and the effectiveness of an adversarial defense should be tested with a variety of algorithms, including the BOBYQA based algorithm further developed in this paper.

The outline of the paper is as follows: in Sect. 2 we present how an adversarial example is generated by solving an optimization problem, and how DFO methods fit in this context. For completeness, we also summarize the model-based BOBYQUA method in Sect. 2.2 as the manuscript (Ughi et al. 2019) where it was introduced for adversarial misclassification is unpublished. In Sect. 3 we present two popular techniques used in existing methods to improve the efficiency and scalability to high dimensional inputs. Section 4 presents the experimental setup and a comparative analysis of existing algorithms along with a focus on our proposed BOBYQA based algorithm. We close with some concluding remarks in Sect. 5.

2 Adversarial examples formulated as an optimization problem

In classification tasks, a DNN outputs a vector whose length is equal to the number of classes and the DNN parameters are trained to match the maximum element of the given output to the correct class of the input. Adversarial perturbations are obtained by modifying the input in such a way that the maximum element of DNN output corresponds to a target class different from the original one.

Consider a classification operator $F : \mathscr{X} \rightarrow \mathscr{C}$ from input space $\mathscr{X}$ to output space $\mathscr{C}$ of classes. A targeted adversarial perturbation $\boldsymbol{\eta}$ to an input $\mathbf{X} \in \mathscr{X}$ has the property that it changes the classification to a specified target class t, i.e $F(\mathbf{X}) = c$ and $F(\mathbf{X} + \boldsymbol{\eta}) = t \neq c$.

Following the formulation in (Alzantot et al. 2019); given an input space $\mathscr{X} = [l, u]^n$, with l and u being respectively the minimum and maximum values of the interval in which the pixels may vary, an output space $\mathscr{C} = \{1, \dots, n_c\}$, where n_c is the number of classes, a maximum energy budget ε_∞, and a suitable loss function $\mathscr{L}$, then the task of computing the adversarial perturbation $\boldsymbol{\eta}$ can be cast as an optimization problem such as

$$\begin{aligned} \min_{\boldsymbol{\eta}} \ & \mathscr{L}(\mathbf{X}, \boldsymbol{\eta}) \\ \text{s.t.} \ & \|\boldsymbol{\eta}\|_\infty \leq \varepsilon_\infty; \\ & [\mathbf{X} + \boldsymbol{\eta}]_j \geq l \qquad \forall j \in 1, \dots, n \\ & [\mathbf{X} + \boldsymbol{\eta}]_j \leq u \qquad \forall j \in 1, \dots, n \end{aligned} \tag{1}$$

where the final two inequality constraints are due to the perturbed image being still an image, i.e. $(\mathbf{X} + \boldsymbol{\eta}) \in \mathscr{X}$. Denoting the pre-classification output vector by $f(\mathbf{X})$, i.e. $F(\mathbf{X}) = \arg\max f(\mathbf{X})$, then the misclassification of $\mathbf{X}$ to target label t is achieved by $\boldsymbol{\eta}$ if $f(\mathbf{X} + \boldsymbol{\eta})_t \geq \max_{j \neq t} f(\mathbf{X} + \boldsymbol{\eta})_j$. As demonstrated in Alzantot et al. (2019), Carlini and Wagner (2017), Chen et al. (2017), in this study we consider the following loss function for computing $\boldsymbol{\eta}$ in (1)

$$\mathscr{L}(\mathbf{X}, \boldsymbol{\eta}) = \log\left(\Sigma_{j \neq t} f(\mathbf{X} + \boldsymbol{\eta})_j\right) - \log\left(f(\mathbf{X} + \boldsymbol{\eta})_t\right). \tag{2}$$

Not having access to the internal parameters of the DNN, the gradient of the loss over the input space cannot be readily computed and instead the adversarial perturbation is found using specially adapted DFO algorithms.

2.1 Derivative free optimization for adversarial examples

Derivative free optimization is a well developed field with numerous classes of methods, see (Conn et al. 2009) and (Larson et al. 2019) for reviews on DFO principles and algorithms. Example classes of such methods include: direct search methods such as simplex, model-based methods, hybrid methods such as finite differences or implicit filtering, as well as randomized variants of the aforementioned and methods specific to convex or noisy objectives. For the generation of adversarial examples, the algorithms that we consider rely on four types of DFO methods:

- Those where the gradient is computed via finite differences, either by sampling all the canonical directions as in ZOO attack (Chen et al. 2017) or random directions as in the Frank-Wolfe algorithm (Chen et al. 2020);
- Those where the solution is thought to be in one of the vertices of the ℓ^∞ domain, i.e. $\boldsymbol{\eta}_i \in \{-\varepsilon_\infty, \varepsilon_\infty\}$ for any i. The Parsimonious algorithm (Moon et al. 2019) implements a combinatorial direct-search within the different possible vertices, initializing the perturbation to $-\varepsilon_\infty$ for all the pixels and then switching collections of them to $+\varepsilon_\infty$, when such an action decreases the loss function. The Square algorithm (Andriushchenko et al. 2020) instead implements a randomized direct-search method where square blocks of pixels are iteratively perturbed to be either $+\varepsilon_\infty$ or $-\varepsilon_\infty$;
- Those where a direct search over the perturbation domain is performed using a genetic method such as GenAttack (Alzantot et al. 2019).
- Those referred to as model-based methods, where the loss function (1) is approximated from its samples with a continuous function which is then minimized using techniques from continuous optimization. *Bounded Optimization BY Quadratic Approximation* (BOBYQA) (Powell 2009) is the first such model-based method adapted to generate adversarial examples in the workshop manuscript (Ughi et al. 2019), motivated by its efficacy in climate modelling (Tett et al. 2013) where the aim is to minimize the number of function samples required. As Ughi et al. (2019) is unpublished, we describe model-based methods in more detail in Subsect. 2.2 for completeness, and state some improvements of Ughi et al. (2019) in Subsect. 4.1.

2.2 Model-based DFO

Given a set of q samples $\mathscr{Y} = \{\mathbf{y}^1, ..., \mathbf{y}^q\}$ with $\mathbf{y}^i \in \mathbb{R}^n$, model-based DFO methods start by identifying the minimizer of the objective among the samples at iteration k, $\mathbf{x}^k = \arg\min_{\mathbf{y}\in\mathscr{Y}} \mathscr{L}(\mathbf{y})$. Following this, a model for the objective function $\mathscr{L}$ is constructed, typically centered around the minimizer. In its simplest form one uses a polynomial approximation to the objective, such as a quadratic model centered in $\mathbf{x}^k$

$$m_k(\mathbf{x}^k + \mathbf{p}) = a_k + \mathbf{c}_k^\top \mathbf{p} + \frac{1}{2}\mathbf{p}^\top \mathbf{M}_k \mathbf{p}, \tag{3}$$

with $a_k \in \mathbb{R}$, $\mathbf{c}_k$, $\mathbf{p} \in \mathbb{R}^n$, and $\mathbf{M}_k \in \mathbb{R}^{n\times n}$ being also symmetric. In a white-box setting one would set $\mathbf{c}_k = \nabla \mathscr{L}(\mathbf{x}^k)$ and $\mathbf{M}_k = \nabla^2 \mathscr{L}(\mathbf{x}^k)$, but this is not feasible in the black-box setting as we do not have access to the derivatives of the objective function. Thus at each iteration k, the parameters a_k, $\mathbf{c}_k$ and $\mathbf{M}_k$ are usually defined by imposing interpolation conditions

$$m_k(\mathbf{y}^i) = \mathscr{L}(\mathbf{y}^i) \quad \forall i \in 1, 2, \dots, q, \tag{4}$$

and when $n + 1 \leq q < 1 + n + n(n+1)/2$ (i.e. the system of equations is underdetermined) the model could be set as linear by imposing $\mathbf{M}_k = \mathbf{0}$ for any k (Nocedal and Wright 2006), or the interpolation conditions could be considered as the constraint in an optimisation problem, as done in the BOBYQA method by (Powell 2009) presented in the following subsection. The objective model (3) is considered to be a good estimate of the objective in a neighborhood referred to as a trust region. Once the model m_k is generated, the update step $\mathbf{p}$ is computed by solving the trust region problem

$$\begin{aligned} \min_{\mathbf{p}} \quad & m_k(\mathbf{x}_k + \mathbf{p}) \\ \text{s.t.} \quad & \|\mathbf{p}\| \leq \Delta, \end{aligned} \tag{5}$$

where Δ is the radius of the region where we believe the model to be accurate, for more details on trust region methods see (Nocedal and Wright 2006). The new point $\mathbf{x}_k + \mathbf{p}$ is added to $\mathscr{Y}$ and a prior point is potentially removed to improve the accuracy of the model according to geometric considerations, such as the poisedness of the sample set which minimizes the potential for degeneracy of the model, (Scheinberg and Toint 2010) for details. In this paper, we consider an exemplary model-based method called BOBYQA.

2.2.1 BOBYQA

The *Bound Optimization BY Quadratic Approximation* (BOBYQA) method, introduced in Powell (2009), updates the parameters of the model $a, \mathbf{c}$, and $\mathbf{M}$, in each iteration in such a way as to minimize the change in the quadratic term $\mathbf{M}_k$ between iterates while otherwise fitting the sample values:

$$\begin{aligned} &\min_{a_k, \mathbf{c}_k, \mathbf{M}_k} \|\mathbf{M}_k - \mathbf{M}_{k-1}\|_F^2 \\ &\text{s.t.} \quad m_k(\mathbf{y}^i) = \mathscr{L}(\mathbf{y}^i), \qquad \forall i \in 1, 2, \dots, q, \end{aligned} \tag{6}$$

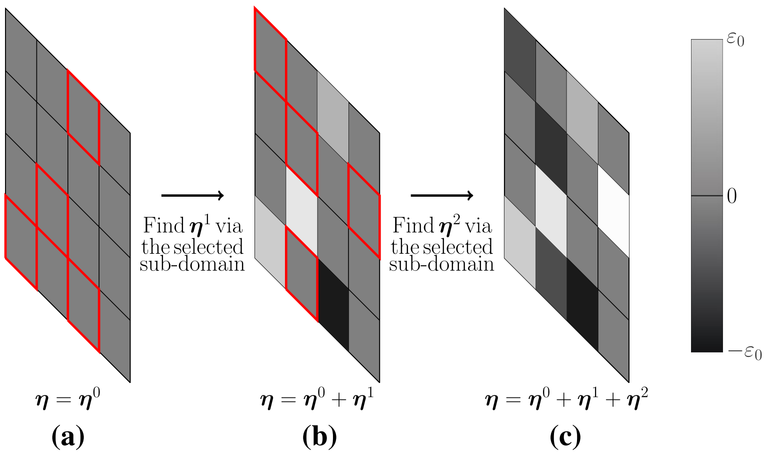
with $n + 1 < q < 1 + n + n(n + 1)/2$ and $\mathbf{M}_k$ initialized as the zero matrix. When the number of parameters $q = n + 1$ then the model is considered as linear with $\mathbf{M}_k$ set as zero. At each iteration the set $\mathscr{Y}$ is updated with the insertion of the new point, $\mathbf{x}_k + \mathbf{p}$, and the removal of the sample which affects the most negatively the model accuracy. The two main notions used to determine which sample to remove are distance from the new sample and a measure to minimize the potential for degeneracy of the model, considerations that are built on concepts of stability (Powell 2009) and of poisedness (Scheinberg and Toint 2010), for more details refer to (Roberts 2019, Chapter 6). This process of maintaining a fixed number of samples insures that the dimension of $\mathscr{Y}$ is fixed.

3 Improving efficiency and computational scalability

Because of the high number of pixels in the input images, the generation of adversarial examples involves solving a high dimensional problem, which makes the use of any DFO method impractical; for instance, the application of the BOBYQA method requires the solution of (6) which scales in memory allocation at least quadratically with the input dimension, and thus is computationally too expensive. Consequently, the implementation of DFO based adversarial algorithms relies on strategies to reduce the dimensionality of the problem, this improves the computational scalability along with the efficiency, as demonstrated experimentally. Instead of solving (1) for $\boldsymbol{\eta} \in \mathbb{R}^n$ directly, the DFO based algorithms consider variations of the domain sub-sampling and/or the hierarchical liftings techniques. Domain sub-sampling iteratively sweeps over batches of $b \ll n$ variables, while hierarchical lifting clusters and perturbs variables simultaneously, as described in following sections.

3.1 Domain sub-sampling

The simplest version of domain sub-sampling consists of partitioning the input dimension into smaller disjoint domains and optimizing the loss function in each of them sequentially. That is, in an n dimensional problem, one considers $k = \lceil n/b \rceil$ sets of integers, $\{\Omega^j\}_{j=1}^k$, of size $b \ll n$ which are disjoint and which cover all of $[n]$. Then (1) is solved sequentially on the dimensions identified by the sets Ω^j. This is possible since the optimization domain is box like, i.e. $\boldsymbol{\eta} \in [l, u]^n$, and each dimension's bound is independent from the others. Formally, rather than solving (1) for $\boldsymbol{\eta} \in \mathbb{R}^n$ directly, for each of $j = 1, \dots, k$ one *sequentially* solves for the $\boldsymbol{\eta}^j \in \mathbb{R}^n$ variables which are only non-zero for entries in Ω^j. The resulting sub-domain perturbations $\boldsymbol{\eta}^j$ are then summed to generate the full perturbation $\boldsymbol{\eta} = \sum_{j=1}^k \boldsymbol{\eta}^j$, see Fig. 2 as an example. That is, the optimization problem (1) is adapted to repeatedly looping over $j = 1, \dots, k$:



Find $\boldsymbol{\eta}^1$ via the selected sub-domain
Find $\boldsymbol{\eta}^2$ via the selected sub-domain
ε_0
0
$-\varepsilon_0$
$\boldsymbol{\eta} = \boldsymbol{\eta}^0$
$\boldsymbol{\eta} = \boldsymbol{\eta}^0 + \boldsymbol{\eta}^1$
$\boldsymbol{\eta} = \boldsymbol{\eta}^0 + \boldsymbol{\eta}^1 + \boldsymbol{\eta}^2$
(a)
(b)
(c)

Fig. 2 Example of how the perturbation $\boldsymbol{\eta}$ evolves through the iterations when an image in $\mathbb{R}^{4\times4}$ is attacked. In **a** the perturbation is $\boldsymbol{\eta} = \boldsymbol{\eta}^0$ and a sub-domain of $b = 4$ pixels (in red) is selected. Once the optimal perturbation $\boldsymbol{\eta}^1$ in the selected sub-domain is found, the perturbation is updated in **b** and a new sub-domain of dimension b is selected. The same is repeated in **c**

$$
\begin{aligned}
\min_{\boldsymbol{\eta}^j} \quad & \mathscr{L}\left(\mathbf{X} + \sum_{h\neq j} \boldsymbol{\eta}^{\ell}, \boldsymbol{\eta}^j\right) \\
\text{s.t.} \quad & \left\| \sum_{h=1}^{k} \boldsymbol{\eta}^h \right\|_\infty \leq \varepsilon_\infty; \\
& \left[\mathbf{X} + \sum_{h=1}^{k} \boldsymbol{\eta}^h\right]_r \geq l \qquad \forall r \in \Omega^j; \\
& \left[\mathbf{X} + \sum_{h=1}^{k} \boldsymbol{\eta}^h\right]_r \leq u \qquad \forall r \in \Omega^j,
\end{aligned}
\tag{7}
$$

where the sets $\{\Omega^j\}_{j=1}^k$ are usually computed again once j is equal to k, and the sub-domain perturbations $\boldsymbol{\eta}^j$ are initialized as null.

We identified three possible ways of selecting the sub-domains $\{\Omega^j\}_{j=1}^k$;

- In *Random Sampling* one considers at each iteration a different random sub-samplings of the domain, i.e. $k = 1$. The ZOO algorithm used this kind of sampling (Chen et al. 2017).
- In *Ordered Sampling* one generates a random disjoint partitioning of the domain, i.e. $k = \lceil n/b \rceil$ and $\Omega_j \cap \Omega_l = \emptyset$ for any j and l. A new partitioning is generated when each variable has been optimized over once. This sampling is implemented in the Parsimonious algorithm.
- In *Variance Sampling* one still generates a random disjoint partitioning of the domain, but chooses the sub-sampling sets $\{\Omega^j\}_{j=1}^k$ in order to optimize over the

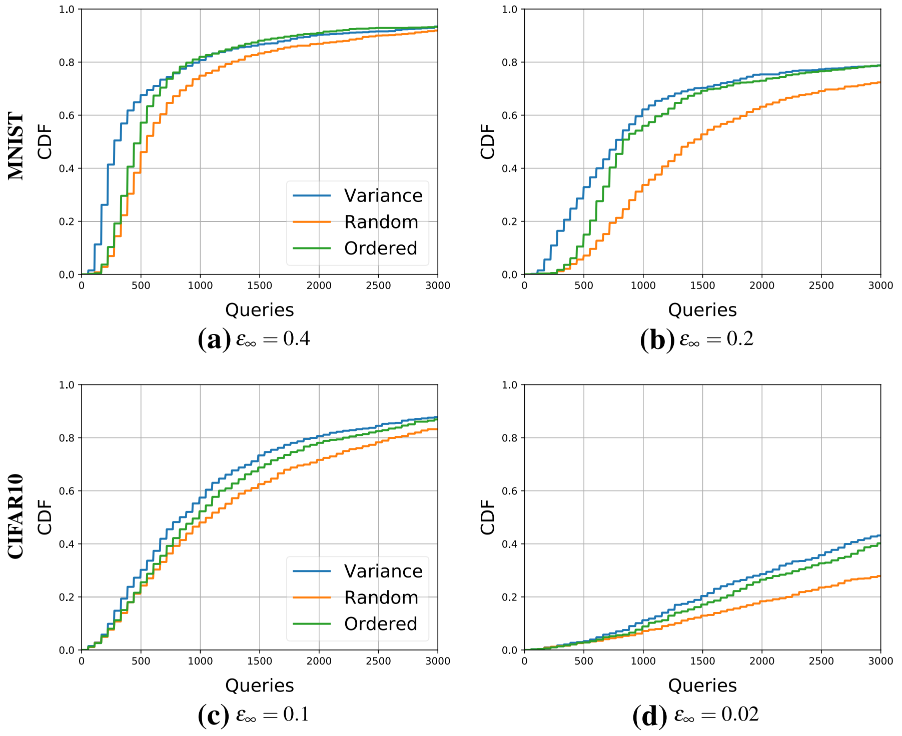


Fig. 3 Cumulative distribution function of successfully perturbed images as a function of number of queries by the BOBYQA based algorithm attacking DNNs trained on the MNIST and the CIFAR10 datasets. In each image the effectiveness of different sub-sampling methods in generating a successful adversarial example is shown for different values of maximum perturbation energies ε_∞. See (Ughi et al. 2019) for details about experimental setup

dimensions that have highest local variance in intensity first. Specifically, the variables are ordered by the variance in intensity among the 8 neighboring variables (e.g. pixels) in the same color channel of the input $\mathbf{X}$. The sets $\{\Omega^j\}_{j=1}^k$ are further reinitialized after each loop through $j = 1, \ldots, k$.

The sub-sampling of the domain affects the efficiency with which an algorithm successfully finds an adversarial example. For instance, in Fig. 3 we compare how these different sub-sampling techniques affect the BOBYQA based algorithm when generating adversarial example for the MNIST and CIFAR10 dataset. It can be observed that variance sampling consistently has a higher success rate cumulative distribution function as compared with random and ordered sampling. This suggest that pixels belonging to high-contrast regions are more influential than the ones in low-contrast ones, and hence variance sampling is the preferable ordering.

To simplify the notation in the following section, the optimization variable is considered to be $\boldsymbol{\eta}^j = \boldsymbol{\Omega}^j \tilde{\boldsymbol{\eta}}^j$ where $\tilde{\boldsymbol{\eta}}^j \in \mathbb{R}^b$ and $\boldsymbol{\Omega}^j \in \mathbb{R}^{n \times b}$ is such that $[\boldsymbol{\Omega}^j]_{pq}$ is one

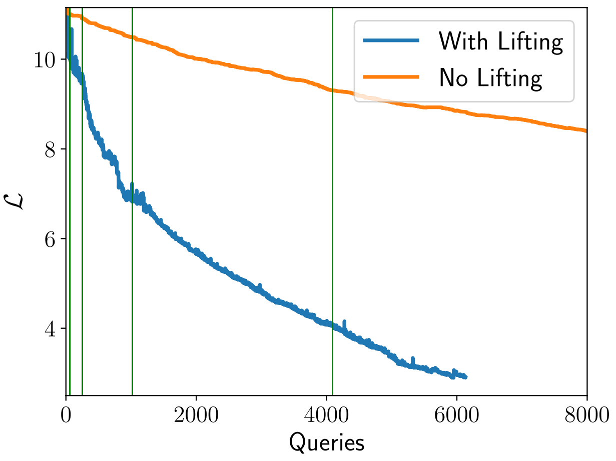


Fig. 4 Impact of hierarchical lifting approach on Loss function (2) as a function of the number of queries to a ResNet50 trained on ImageNet dataset to find the adversarial example for a single image with the BOBYQA based method. The green vertical lines correspond to changes of hierarchical level, which entail an increase in the dimension of the optimization space

if the qth element of Ω^j is p, zero otherwise. The implementation of variance sampling method at iteration j in a domain of dimension n_ℓ is summarized in Algorithm 1.

Algorithm 1 GENERATE_SAMPLING_MATRIX($\hat{\mathbf{X}}$,n_ℓ,b,j)

```
1: $\boldsymbol{\Omega} \leftarrow \mathbf{0} \in \mathbb{R}^{n_\ell \times b}$, $\boldsymbol{\mu} \leftarrow \mathbf{0} \in \mathbb{R}^{n_\ell}$,$\boldsymbol{v} \leftarrow \mathbf{0} \in \mathbb{R}^{n_\ell}$.
2: for $i = 1, \dots, b$ do
3:     $\boldsymbol{\mu}(i) = \sum_{j \in \text{Neighbours}(i)} \hat{\mathbf{X}}(j)$.
4:     $\mathbf{v}(i) = \sum_{j \in \text{Neighbours}(i)} \hat{\mathbf{X}}(j)^2 - \boldsymbol{\mu}^2(i)$.    # computation of the variance around the variable.
5: end for
6: $\mathbf{s} \leftarrow$ argsort $\mathbf{v}$.
7: for $i = 1, \dots, b$ do
8:     $\boldsymbol{\Omega}(\mathbf{s}[i + j \times b], [i]) = 1$.
9: end for
10: Return $\boldsymbol{\Omega}$.
```

3.2 Hierarchical lifting

Authors of ZOO attack (Chen et al. 2017) demonstrated that fewer queries are required to find adversarial example when pixels are considered in clusters, and not independently. This lead to the hierarchical lifting approach where one optimizes over increasingly higher dimensional spaces at each step, referred here as level ℓ; Figure 4 shows how effective this approach is when implementing the BOBYQA based algorithm. These low dimensional spaces are lifted to the image space via a

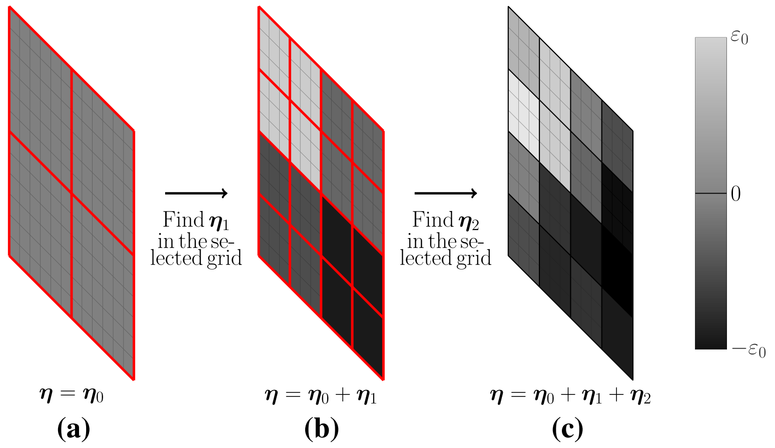

Find η_1 in the selected grid
Find η_2 in the selected grid
ε_0
0
$-\varepsilon_0$
$\eta = \eta_0$
(a)
$\eta = \eta_0 + \eta_1$
(b)
$\eta = \eta_0 + \eta_1 + \eta_2$
(c)

Fig. 5 Example of how the perturbation $\boldsymbol{\eta}$ is generated in a hierarchical lifting method with $n_1 = 4$ and $n_2 = 16$ on an image in $\mathbb{R}^{12\times 12}$. In **a** the perturbation is $\boldsymbol{\eta} = \boldsymbol{\eta}_0$ and the boxes generated via the grid of dimension n_1 are highlighted in red. Once the optimal perturbation $\boldsymbol{\eta}_1$ is found, the perturbation is updated in **b** and the image is further divided with a grid with n_2 blocks. The final solution obtained after optimization is shown in **c**

linear lifting, where at each level ℓ a linear lifting $\mathbf{D}^\ell : \mathbb{R}^{n_\ell} \to \mathbb{R}^n$ is considered and a perturbation $\hat{\boldsymbol{\eta}}_\ell \in \mathbb{R}^{n_\ell}$ is found to be added to the full perturbation $\boldsymbol{\eta}$, according to

$$\boldsymbol{\eta} = \sum_{j=0}^{\ell} \boldsymbol{\eta}_j = \sum_{j=0}^{\ell} \mathbf{D}^j \hat{\boldsymbol{\eta}}_j. \tag{8}$$

Here $\boldsymbol{\eta}_0$ is initialized as $\underline{\mathbf{0}}$ and the perturbations $\boldsymbol{\eta}_j$ of the previous layers are considered as fixed. An example of how this works is illustrated in Fig. 5. The hierarchical lifting considered here is analogous to the derivative-based Recursive Multiscale Trust-Region method in Gratton et al. (2008). Our piece-wise constant lifting is substantially simpler than (Gratton et al. 2008) in that we only progress from coarse to fine grids as opposed to “W” and “V” cycles between scales; this simplicity is beneficial for the misclassification application here where our aim is to minimize the number of function queries used by the DFO method.

All the methods considered in this work rely on ideas which can be interpreted through this approach. The algorithms that we consider rely on two kinds of linear lifting $\mathbf{D}^\ell$ differentiated by the way each scalar in $\hat{\boldsymbol{\eta}}$ is associated to a set of pixels in the original image domain $\mathbb{R}^n$; namely the random and the block liftings. The former relates a random set of pixels of the original image to each hyper-variable; this forces the perturbation to be of high-frequency nature, as illustrated in Fig. 6a, which several articles indicate as being the most effective (Guo et al. 2018; Gopalakrishnan et al. 2018; Sharma et al. 2019). The GenAttack and Frank-Wolfe algorithms use a variation of this kind of lifting. The latter instead is based on interpolation operations; a sorting matrix $\mathbf{S}^\ell : \mathbb{R}^{n_\ell} \to \mathbb{R}^n$ is applied such that every index of $\hat{\boldsymbol{\eta}}_\ell$ is uniquely associated to a node of a coarse grid masked over the original image. Afterwards, an interpolation $\mathbf{L}^\ell : \mathbb{R}^n \to \mathbb{R}^n$ is implemented over the values in the coarse grid, i.e.

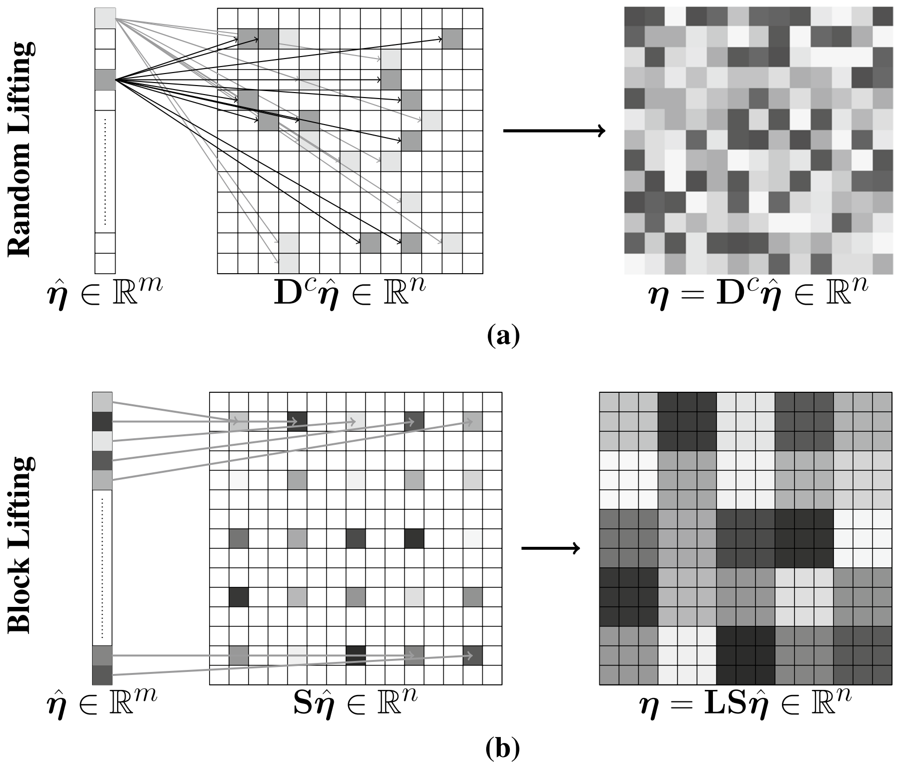

Random Lifting
$\hat{\boldsymbol{\eta}} \in \mathbb{R}^m$
$\mathbf{D}^c\hat{\boldsymbol{\eta}} \in \mathbb{R}^n$
$\boldsymbol{\eta} = \mathbf{D}^c\hat{\boldsymbol{\eta}} \in \mathbb{R}^n$
(a)
Block Lifting
$\hat{\boldsymbol{\eta}} \in \mathbb{R}^m$
$\mathbf{S}\hat{\boldsymbol{\eta}} \in \mathbb{R}^n$
$\boldsymbol{\eta} = \mathbf{L}\mathbf{S}\hat{\boldsymbol{\eta}} \in \mathbb{R}^n$
(b)

Fig. 6 Examples for **a** random and **b** block liftings. In the random case each pixel in the perturbation is associated to just one element of $\hat{\boldsymbol{\eta}}_\ell$. Block lifting uses a piece-wise constant interpolation $\mathbf{L}$ over a coarse grid $\mathbf{S}\hat{\boldsymbol{\eta}}_\ell$ and each block is associated uniquely to one of the variables in $\hat{\boldsymbol{\eta}}_\ell$. In both cases, the lifting $\mathbf{D}$ is such that each element $\mathbf{D}_{ij}$ is either 1 or 0

$\boldsymbol{\eta}_\ell = \mathbf{L}^\ell\mathbf{S}^\ell\hat{\boldsymbol{\eta}}_\ell = \mathbf{D}^\ell\hat{\boldsymbol{\eta}}_\ell$. Both Square and Parsimonious algorithms implement hierarchical lifting with the piece-wise constant interpolation, here referred to as block lifting. At the lower levels, the interpolation lifting generates low frequency perturbations, as illustrated in Fig. 6b.

Since n_ℓ may still be very high, for each level ℓ domain sub-sampling is also applied considering $\hat{\boldsymbol{\eta}}_\ell = \sum_{j=0}^{k}\tilde{\boldsymbol{\eta}}_\ell^j$. In the piece-wise constant case with variance sampling, the blocks are ordered according to the variance of mean intensity among neighboring blocks, in contrast to the variance within each block as suggested in Chen et al. (2017). Consequently, at each level the adversarial example is found by solving the following iterative problem

$$
\begin{aligned}
&\min_{\tilde{\boldsymbol{\eta}}_\ell^j} \mathscr{L}\left(\mathbf{X}+\bar{\boldsymbol{\eta}}, \mathbf{D}^\ell \boldsymbol{\Omega}^k \tilde{\boldsymbol{\eta}}_\ell^j\right) \\
\text{s.t.} \quad & \left\|\bar{\boldsymbol{\eta}}+\mathbf{D}^\ell \boldsymbol{\Omega}^k \tilde{\boldsymbol{\eta}}_\ell^j\right\|_\infty \leq \varepsilon_\infty \\
& \left[\mathbf{X}+\bar{\boldsymbol{\eta}}+\mathbf{D}^\ell \boldsymbol{\Omega}^k \tilde{\boldsymbol{\eta}}_\ell^j\right]_r \geq l \qquad \forall r \in \{1, \ldots, n\} \\
& \left[\mathbf{X}+\bar{\boldsymbol{\eta}}+\mathbf{D}^\ell \boldsymbol{\Omega}^k \tilde{\boldsymbol{\eta}}_\ell^j\right]_r \leq u \qquad \forall r \in \{1, \ldots, n\},
\end{aligned} \tag{9}
$$

where $\bar{\boldsymbol{\eta}} = \sum_{i=0}^{\ell-1} \boldsymbol{\eta}_i + \mathbf{D}^\ell \sum_{m \neq j} \hat{\boldsymbol{\eta}}_\ell^m$. Algorithm 2 gives an implementation of the block lifting matrix when in the grid has dimension n_ℓ.

Algorithm 2 GENERATE_LIFTING(n_ℓ,n)

```
1: $\mathbf{D} \leftarrow \mathbf{0} \in \mathbb{R}^{n \times n_\ell}$
2: for $i = 1, \ldots, n_\ell$ do
3:     Generate set of pixels $S$ that are in the block associated to the $i$-th element of the $n_\ell$ dimensional
       super-grid.
4:     for $j \in S$ do
5:         $\mathbf{D}(i, j) = 1$.
6:     end for
7: end for
8: Return $\mathbf{D}$.
```

4 Comparison of derivative free methods

In this section, we compare algorithms based on a selection of state-of-the-art DFO methods. In particular we consider an improved version of the BOBYQA based algorithm (Ughi et al. 2019), GenAttack algorithm (Alzantot et al. 2019), Parsimonious algorithm (Moon et al. 2019), Square algorithm (Andriushchenko et al. 2020) and Frank-Wolfe algorithm (Chen et al. 2020) in the following two frameworks:

– Section 4.3 considers the canonical setup for black-box adversarial attacks on which the considered algorithms have been tuned in their respective articles. Specifically, we consider attacks on networks trained adversarially or not on CIFAR10 and ImageNet, two popular datasets in the literature, and with no further defense implemented.
– Section 4.4 considers a setup that simulates structural defenses on which the different algorithms were not tuned. We limit the perturbation to a fixed number of pixels with high variance in intensity considering attacks on a network non-adversarially trained on the CIFAR10 dataset.

The performance of all algorithms is measured in terms of the distribution of queries needed to successfully find adversaries to identical networks given a fixed ℓ^∞ perturbation constraint and the same input images. In particular, the algorithms are compared according to the cumulative fraction of images successfully misclassified (abridged by CDF for cumulative distribution function) as a function of the number of queries to the DNN, which corresponds to the data profile comparison measure introduced in Moré and Wild (2009). For each experimental setting $\mathscr{A}$, the single attacks are denoted by a, and the following variable is introduced

$$t_a = \# \text{ of queries to find an adversarial attack} \tag{10}$$

that is set to infinity in case the adversarial example is not found. Thus, the CDF for a number of queries α is

$$CDF(\alpha) = \frac{1}{|\mathscr{A}|} size\{a \in \mathscr{A} : t_a \leq \alpha\}. \tag{11}$$

4.1 Parameter setup for algorithms

The experiments use publicly available implementations for the GenAttack (Alzantot et al. 2019), Parsimonious (Moon et al. 2019), Square (Andriushchenko et al. 2020), and Frank-Wolfe (Chen et al. 2020) algorithms[1] using the same hyper-parameter setting and hierarchical lifting approach as suggested by the respective authors.

Following (Ughi et al. 2019), for the BOBYQA based algorithm we consider linear models to approximate the loss function; i.e., $\mathbf{M} = \mathbf{0}$ and $q = n + 1$ at all iterations. Further, we use the variance sub-sampling method as done in (Ughi et al. 2019). However, here we consider block lifting as described in Sect. 3.2[2], rather than the linear lifting in (Ughi et al. 2019); we consider an initial domain of dimension $n_1 = 2 \times 2 \times 3$, and double the refinement of the grid at each layer, i.e. $n_{\ell+1} = 4n_\ell$; we set the batch sampling size $b = 25$. The BOBYQA based algorithm is summarized in Algorithm 3 and a Python implementation of the proposed algorithm based on BOBYQA package from Cartis et al. (2019) is available on Github[3].

[1] GenAttack: https://github.com/nesl/adversarial_genattack.
Parsimonious algorithm: https://github.com/snu-mllab/parsimonious-blackbox-attack.
Square algorithm: https://github.com/max-andr/square-attack.
Frank-Wolfe algorithm https://github.com/uclaml/Frank-Wolfe-AdvML.

[2] The choice for this kind of lifting was driven by preliminary experiments in which we considered also a grid method with linear interpolation and a random lifting method as well. It is possible to run the analysis using the code in [3]

[3] https://github.com/giughi/An-Empirical-Study-of-DFO-Algorithms-for-Targeted-Black-Box-Attacks-in-DNNs

Algorithm 3 BOBYQA Based Algorithm

```
1: Input: Image $\mathbf{X} \in \mathbb{R}^n$, target label $t$, maximum perturbation $\varepsilon_\infty$, Neural Net $F$, initial hierarchical level
   grid dimensions $m$, maximum number of queries $n^{max}$, batch sampling size $b$, and maximum number $\kappa$ of
   queries that we are allowed to do for each batch.
2: Initialize $\boldsymbol{\eta} \leftarrow \underline{0} \in \mathbb{R}^n$, $n_{eval} = 0$, $\ell = 1$, $n_\ell = 12$.
3: while $\arg\max F(\mathbf{X}+\boldsymbol{\eta}) \neq t$ and $n_{eval} < n^{max}$ do
4:     # Compute the number of sub samplings necessary to cover the whole domain
5:     $num_{sub} = n/(n_\ell * b)$
6:     # Generate the lifting matrix
7:     $\mathbf{D}_\ell$ = GENERATE_LIFTING($n_\ell, n$)
8:     # Minimize on all the sampled sub-domains
9:     for $j = 1, \ldots, num_{sub}$ do
10:        # Compute the matrix which selects b dimensions of the m-dimensional domain.
11:        $\boldsymbol{\Omega}_\ell^j$ = GENERATE_SAMPLING_MATRIX($\mathbf{X}+\boldsymbol{\eta}, n_\ell, b, j$)
12:        # Define the pixel-wise bounds for a perturbation over $\boldsymbol{X}+\boldsymbol{\eta}$.
13:        $\mathbf{a} = \min\{l-\boldsymbol{\eta}, 0\}, \quad \mathbf{b} = \max\{u-\boldsymbol{\eta}, 0\}$
14:        # Find $\hat{\boldsymbol{\eta}}_\ell^j$ by implementing the BOBYQA optimization to the problem (9).
15:        $\hat{\boldsymbol{\eta}}_\ell^j$=BOBYQA($F, \mathbf{X}, \boldsymbol{\eta}, \mathbf{a}, \mathbf{b}, \boldsymbol{\Omega}_\ell^j, \mathbf{D}_\ell, t, \kappa$)    # Algorithm 4
16:        # Update the noise
17:        $\boldsymbol{\eta} += \mathbf{D}_\ell \boldsymbol{\Omega}_\ell^j \hat{\boldsymbol{\eta}}_\ell^j$.
18:        $n_{eval}$ += $\kappa$.
19:    end for
20:    $\ell += 1$, $n_\ell *= 4$.
21: end while
22: if $\arg\max F(\mathbf{X}+\boldsymbol{\eta}) = t$ then
23:    The perturbation is successful.
24: else if $n_{eval} > n^{max}$ then
25:    The perturbation was not successful with $n^{max}$ iterations.
26: end if
```

Algorithm 4 BOBYQA(F, $\mathbf{X}$, $\boldsymbol{\eta}$, $\mathbf{a}$, $\mathbf{b}$, $\boldsymbol{\Omega}_\ell^j$, $\mathbf{D}^\ell$, t, κ)

```
1: # Consider the restricted loss function $\mathscr{L}(\boldsymbol{X}+\boldsymbol{\eta}, \boldsymbol{D}^\ell \boldsymbol{\Omega}_\ell^j(\cdot)) : \mathbb{R}^b \to \mathbb{R}$
2: Build an initial model $m_0$ as in (3) of the loss function based on $b+1$ samples. # The samples consist of the
   initial perturbation $\boldsymbol{X}+\boldsymbol{\eta}$ and the b perturbations obtained by considering changes along the canonical
   directions of $\boldsymbol{x}$ in $\boldsymbol{X}+\boldsymbol{\eta}+\boldsymbol{D}^\ell \boldsymbol{\Omega}_\ell^j \boldsymbol{x}$.
3: Find minimizer $\mathbf{x}$ of $m_o$ such that $\mathbf{D}^\ell \boldsymbol{\Omega}_\ell^j \mathbf{x} \in [\mathbf{a}, \mathbf{b}]$.
4: for $j = 1, \ldots, \kappa - b$ do
5:     Add $\mathbf{x}$ to the set of samples and get rid of the least informative one according to [37].
6:     Build the new model $m_j$ according to (6).
7:     Find minimizer $\mathbf{x}$ of $m_j$ such that $\mathbf{D}^\ell \boldsymbol{\Omega}_\ell^j \mathbf{x} \in [\mathbf{a}, \mathbf{b}]$.
8: end for
9: Return $\mathbf{x}$.
```

4.2 Dataset and neural network specifications

We performed experiments using the popular ResNet50 architecture (He et al. 2016) with two training scenarios; one with the unperturbed images, and one with the defense[4] proposed in Engstrom et al. (2019). The number of experiments and the choice of the targets for each individual dataset is described below.

CIFAR10 The CIFAR10 dataset contains images from 10 classes and of dimension 32x32x3. To generate a comprehensive distribution for the queries at each energy budget, ten correctly classified images are considered per each class, and each of them is targeted to all of the 9 remaining classes; this way we generate a total of 900 attacks per maximum perturbation energy per adversarial method.

ImageNet This dataset contains millions of images with a dimension of 299x299x3 divided among 1000 classes. Because of the high dimensionality and number of classes, random images are attacked considering a random target class. We conducted 200 and 160 tests for networks trained both with and without adversarial training per maximum perturbation energy.

4.3 Results for standard and madryLab Trained DNNs

In Figures 7 and 8 we present the CDF for different maximum perturbation energies ε_∞. The pixels are normalized to be in the interval $(-1/2, 1/2)$, hence, $\varepsilon_\infty = 0.1$ would imply that any pixel is allowed to change 10% of the total intensity range from its initial value. The CDFs are illustrated so that we can easily see which method has been able to misclassify the largest fraction of images in the given test-set for a fixed number of queries to the DNN. The confidence intervals of the CDFs are reported in Appendix 1 and they entail that the CDFs identify the best algorithms in the CIFAR10 case almost surely while in the ImageNet one with high confidence.

For the CIFAR10 dataset in Figure 7, we observe that algorithms that search the perturbation directly in the vertices of the perturbation domain require the least amount of network queries. In the case of non-adversarially trained networks, the Square algorithm is able to misclassify using the least number of queries; this is demonstrated by its associated solid green CDF being consistently above that of the other methods. Specifically, when $\varepsilon_\infty = 0.05$, at 1000 queries Square algorithms has a CDF of 0.97 compared to 0.94 and 0.88 of the Parsimonious and BOBYQA methods respectively, and for $\varepsilon_\infty = 0.005$ at 3000 queries Square achieves a CDF of 0.20 which is 50% times higher than Parsimonious and BOBYQA. When the net is instead trained adversarially, dashed lines, Square algorithm looses a lot of its effectiveness becoming comparable to the BOBYQA based method, while Parismonious algorithm achieves almost always the highest fraction of successfully perturbed images for any given maximum number of queries. For example, when $\varepsilon_\infty = 0.05$ at 3,000 queries the CDF of Parisomonious is 0.29 compared to 0.25 and 0.23 of Square and BOBYQA.

In the ImageNet dataset, see Figure 8(a), we observe that an adversarial method can be especially susceptible to particular defenses. Specifically, when the network is trained without a defense, the Square algorithm has a success rate CDF that is consistently higher than the other methods, but the success rate CDF for the Square

[4] These networks are available already trained at https://github.com/MadryLab/robustness

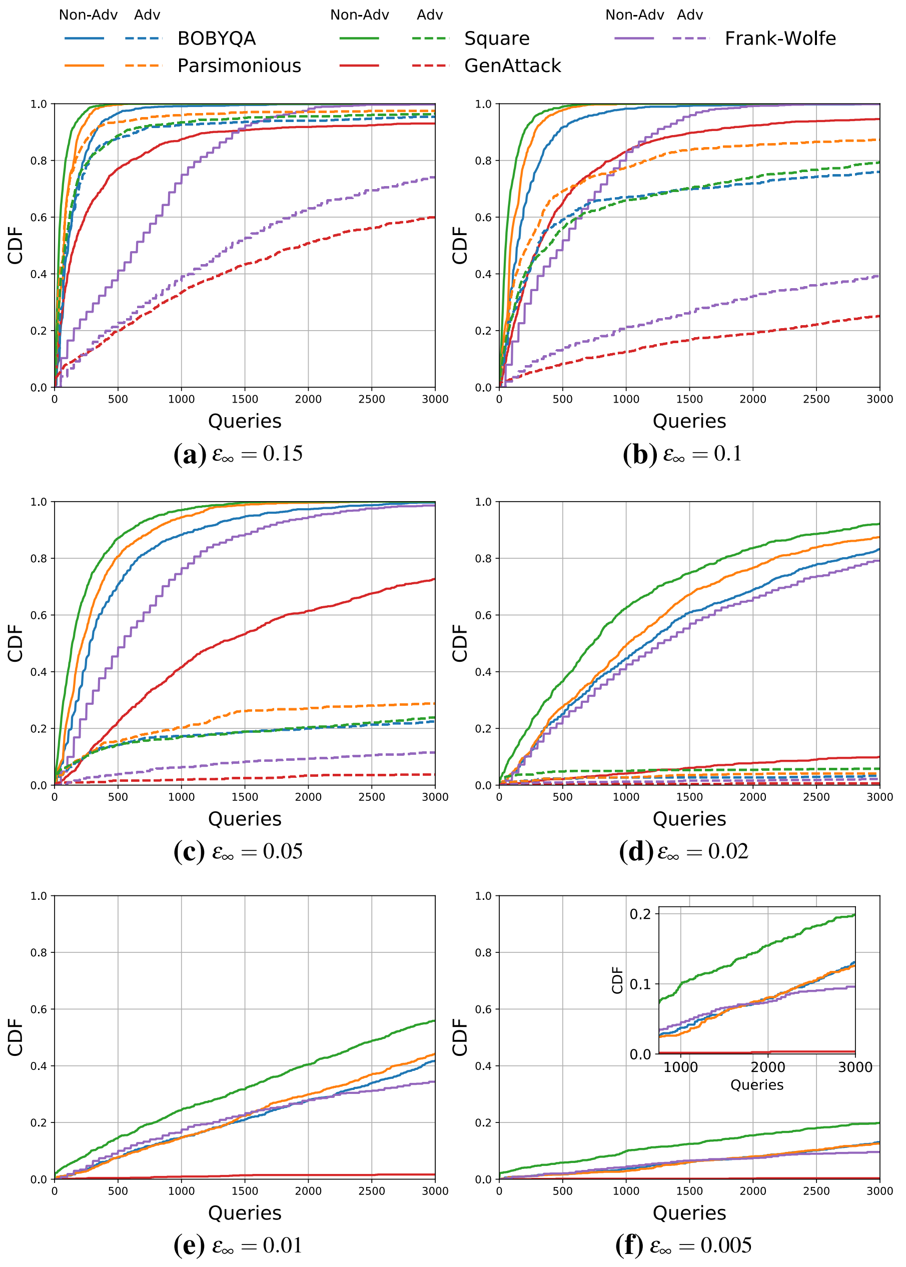


Fig. 7 Cumulative fraction of test set images successfully misclassified with adversarial examples generated by GenAttack, Parsimonious, Square, Frank-Wolfe, and our BOBYQA based approaches for different maximum perturbation energies ε_∞ and DNNs trained on the CIFAR10 dataset. In all results the solid and dashed lines denoted by 'Non-Adv' and 'Adv' corresponds to attacks on networks trained without or with the MadryLab defense strategy (Engstrom et al. 2019) respectively

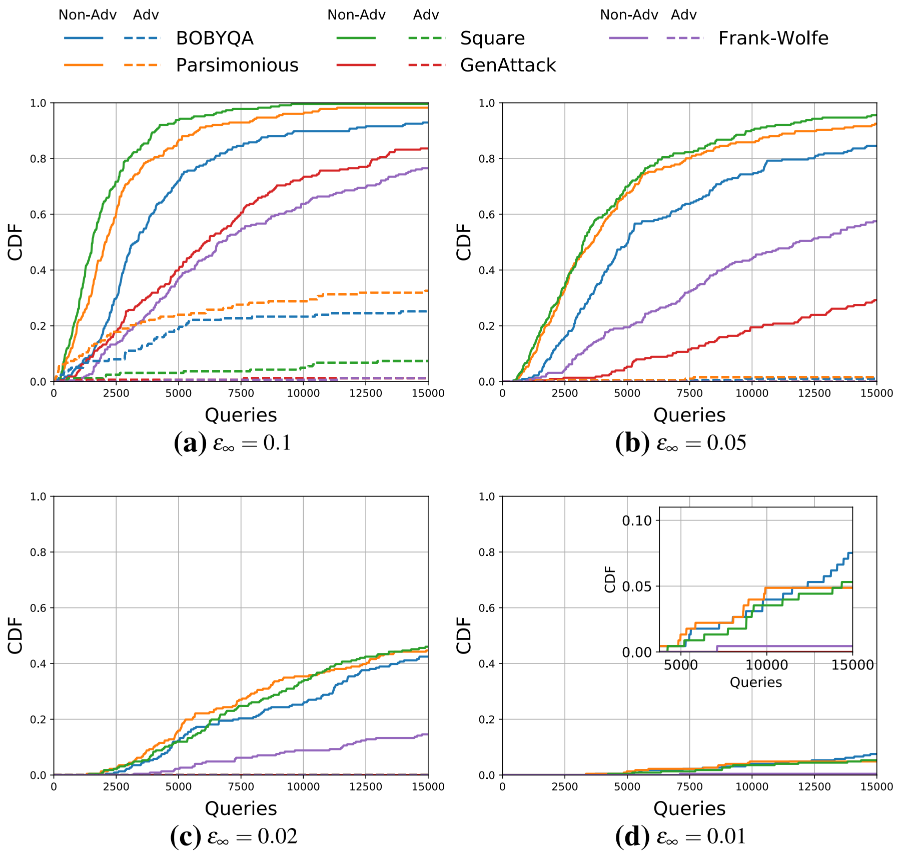


Fig. 8 Cumulative fraction of test set images successfully misclassified with adversarial examples generated by GenAttack, Parsimonious, Square, Frank-Wolfe, and our BOBYQA based approaches for different maximum perturbation energies ε_∞ and DNNs trained on the ImageNet dataset. In all results the solid and dashed lines denoted by 'Non-Adv' and 'Adv' corresponds to attacks on networks trained without or with the MadryLab defense strategy (Engstrom et al. 2019) respectively

algorithm is decreased by the MadryLab defense so that it is substantially less effective than Parsimonious and BOBYQA algorithms. On the other hand, the Parsimonious method achieves similar results to Square algorithm in the non-adversarial case. On average for the different maximum perturbation energies Parsimonious is 0.045 less efficient than Square, but when the defense is introduced it finds the adversarial examples with the least number of queries. In Figure 8(a) Parisomious has a CDF of 0.33 at 15,000 queries while BOBYQA 0.24 and Square 0.07. The rate with which the CDFs decrease as the maximum perturbation energy ε_∞ decreases it also differs by algorithm. The CDF for Square decreases moderately faster than for Parsimonious such that Square has a consistently higher CDF than Parsimonious for $\varepsilon = 0.1$ in Figure 8(a) but consistently lower in Figure 8(d). Moreover, the success rate for BOBYQA decreases the slowest with ε_∞ such that in Figure 8 its CDF is similar to or grater than Parsimonious. Specifically, in Figure 8(d) at 15,000 the final CDF of BOBYQA algorithm queries is 1.42 times higher than the one of the Square algorithm.

The Frank-Wolfe algorithm is able to achieve results comparable to the ones of the methods above while considering the small-dimensional problem of CIFAR10 with a very low maximum perturbation energy. However, when considering the ImageNet case and the adversarially trained DNNs, the Frank-Wolfe algorithm has a substantially lower success rate CDF; e.g. in the ImageNet case with non-adversarial training, Square algorithm achieves a CDF 1.66 times higher than the Frank-Wofle algorithm when $\epsilon_\infty = 0.05$.

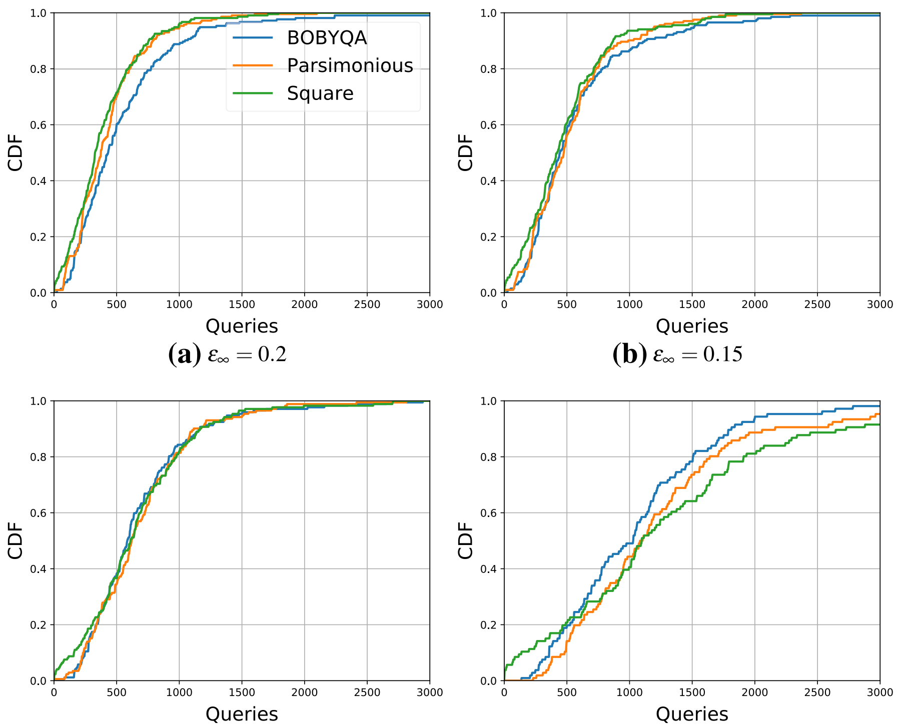
GenAttack has a higher success rate CDF than the Frank-Wolfe algorithm in the ImageNet case for $\epsilon_\infty = 0.1$, see Figure 8(a), but, besides this case, it constantly achieves the lowest success rate.

The relative success of the misclassification algorithms as a function of the allowed perturbation energy is determined by the training loss function and associated partition of the input space into regions associated with each class (Liu et al. 2017)[Figure 3]. Each correctly classified example, not on the boundary between classes, has a small enough ϵ_∞ region surrounding it where misclassification cannot be obtained. As ϵ_∞ is increased the fraction of the perturbations which admit misclassification can be expected to increase, and misclassification becomes trivial once ϵ_∞ is sufficiently large that the majority of vertices correspond to misclassification. In fact, Goodfellow et al. (2015) suggest the pre-classification output vector is maximally misclassified according to (2) at vertices.

For these reasons we can expect that vertex search methods such as Parsimonious and Square are preferable for large ϵ_∞ while model based methods such as BOBYQA and Frank-Wolfe become increasingly beneficial, relatively, as ϵ_∞ decreases and the fraction of vertices which correspond to misclassification becomes small. Although both BOBYQA and Frank-Wolfe are based on a linear approximation of the problem, their respective number of samples taken differs substantially by how the samples are selected to construct the model and the subspaces with which they optimize over. In particular, Frank-Wolfe optimizes over $b = 25$ dimensional subspaces drawn at random from the unit sphere while BOBYQA sequentially optimizes over subsampled batches of variables as described in Sect. 3.1. The relative impact of these differing dimensionality reduction techniques has not been explored and may account for some of the superior performance of BOBYQA as compared to Frank-Wolfe.

4.4 Results with fixed pixel count constraints

In addition to network training designed to increase robustness, such as MadryLab considered previously, there are a multitude of other defenses and real world constraints (Hao-Chen et al. 2020). The relative success rate, or other characteristics, of adversarial algorithms can be expected to differ in these diverse settings. To demonstrate this, we consider one such setting where the maximum number of pixels allowed to be perturbed is limited. This is motivated by the defenses where network inputs are thresholded in a wavelet domain to exclude high frequency perturbations



(a) $\varepsilon_\infty = 0.2$ **(b)** $\varepsilon_\infty = 0.15$

(c) $\varepsilon_\infty = 0.1$ **(d)** $\varepsilon_\infty = 0.05$

Fig. 9 Cumulative fraction of test set images successfully misclassified with adversarial examples generated by Parsimonious, Square, and our BOBYQA based approaches for different maximum perturbation energies ε_∞ against a ResNet50 trained non-adversarially on the CIFAR10 dataset when only the 1000 pixels with the highest variance in intensity in their neighborhood are allowed to be modified

(Guo et al. 2018), as well as by real world constraints such as attacks designed to appear structured such as localized perturbations designed to look like graffiti (Eykholt et al. 2018; Naseer et al. 2019). We allow the algorithms to perturb only the fixed selection of the 1000 pixels of the targeted image that have the highest variance in intensity in their channel neighborhood. Because of the previous results it is possible to identify three methods that work consistently better than the others, and thus only these will be considered, namely: the Parsimonious, the Square, and the BOBYQA based algorithms. To allow the perturbations to be limited to the selected pixels, we consider the Square algorithm with squares of pixel dimension, the Parsimonious algorithm on the finest grid, and the BOBYQA algorithm without the hierarchical lifting, i.e. $\mathbf{D}^1 = \mathbf{I}$ where $\mathbf{I}$ is the identity matrix.

The results reported in Figure 9 suggest that when the domain is dimensionally limited, the most efficient algorithm changes according to the allowed maximum perturbation energy. When the maximum perturbation energy decreases and the linear model is more accurate, the BOBYQA method manages to achieve a higher SR than both Square and Parsimonious algorithms, unlike in the previous experiments. Moreover,

Table 1 Average time required by different algorithms in processing 1000 queries to the ImageNet non-adversarially trained ResNet50

	BOBYQA	GenAttack	Frank-Wolfe	Parismonious	Square
Time/1000 queries	43.7s	11.7s	18.6s	51.8s	12.7s

the Parsimonious algorithm has almost identical behavior to Square algorithm for high energy bounds, but becomes more efficient when the maximum energy is $\varepsilon_\infty = 0.05$. Figures 9 and 7 also differ by the former not employing hierarchical lifting as described in Sect. 3.2 while Figure 7 does make use of lifting. The overall trends between Figures 7 and 9 are consistent which suggests that the use of lifting does not change the overall trends observed between classes of methods, rather it consistently reduces the overall number of samples needed for misclassification.

We also considered experiments on ImageNet, but limiting the number of pixels that could be perturbed did not allow for any successful misclassification with less than 15,000 queries.

4.5 Relative computational cost

While the focus in this manuscript is to compare the different typology of algorithms according to the their misclassification success rate as a function of the number of queries to the DNN; it is also worth noting that the different type of DFO algorithms can be expected to have differing computational burdens. Table 1 displays the average time for each algorithm to update their perturbation of the input per 1,000 queries. In particular, these results are obtained by running 10 attacks to the ResNet50 non-adversarially trained on ImageNet with a perturbation error of $\varepsilon_\infty = 10^{-3}$, and the time was then averaged on one thousand queries. However, we remark that these algorithms were not optimised on a computational point of view and these results are reported mainly for an indicative purpose.

All the algorithms have computational costs on the same order of magnitude. The Square algorithm stands out as having the lowest computational burden and achieving state-of-the-art misclassification rates for non-adversarially trained networks. However, for networks trained with the MadryLab defense, Parsimonious is observed to have a superior misclassification rate, at the cost of taking approximately 4 times longer to compute the perturbation. Finally, the fact that BOBYQA and Parsimonious algorithms are the slowest shows how sophisticated hierarchical approaches are computationally intensive, though can be beneficial for lower perturbation energies or networks trained against adversarial attacks.

5 Discussion and conclusion

We have compared for the first time how the the existing GenAttack (Alzantot et al. 2019), Parsimonious (Moon et al. 2019), Square (Andriushchenko et al. 2020), and Frank-Wolfe (Chen et al. 2020) algorithms, and the herein further developed BOBYQA based method, behave when the available ℓ^∞ energy for a perturbation varies, and an adversarial training or a structural defense is considered.

The results suggest that those methods limiting the search for an adversarial example to the vertices of the ℓ^∞ perturbation domain generally work better. Whilst Square algorithm is especially effective on the non-adversarially trained networks, the Parsimonious algorithm manages to outperform any other approach when the networks are adversarially trained with the MadryLab implementation. Furthermore, the Parsimonious algorithm performs better than Square when considering the structural defense that limits the attacks on some pixels, suggesting that an algorithm based on combinatorial search is robust in its hyper-parameters to the setting where it is applied.

The BOBYQA based algorithm was further developed in this paper to explore how model-based approaches compare to the state-of-the-art algorithms, and was found to achieve similar results to the Parsimonious and Square algorithms. In almost in all the experiments the BOBYQA based algorithm achieves a success rate CDF comparable to the ones of the Parsimonious and the Square algorithms; it achieves the state-of-the-art success rate at saturation for low maximum perturbation energy constraint both in the ImageNet case and in the pixel constrained problem, thus becoming the preferable choice in these cases. Figures 7 and 9 differ in part by the former making use of hierarchical lifting as described in Sect. 3.2 while the later does not employ any lifting. The aforementioned performance trends are consistent in Figures 7 and 9 which suggests that while lifting reduces the overall number of samples, it does not impact the relative performance between algorithms. New dimensionality reduction techniques are a topic of recent investigation for DFO, see for example (Cartis et al. 2020), or variations of the derivative-based multi-level approach (Gratton et al. 2008), and might further improve the results observed here.

In conclusion, we find that both the structure of the algorithm and the attack setting have the potential to impact the algorithm performance. These observations highlight the importance of comparing any new algorithm to the state-of-the-art in a variety of different settings, such as is done here. Similarly, the effectiveness of an adversarial defense for DNNs should always be tested using as wide a range of algorithms as possible.

Appendix A experiments with confidence intervals

The results that are reported in Sect. 4 show how the different algorithms compare in expectation but do not give an estimate on how confident the results are. To get a measure of this, we bootstrap (Efron and Tibshirani 1994) the data. In

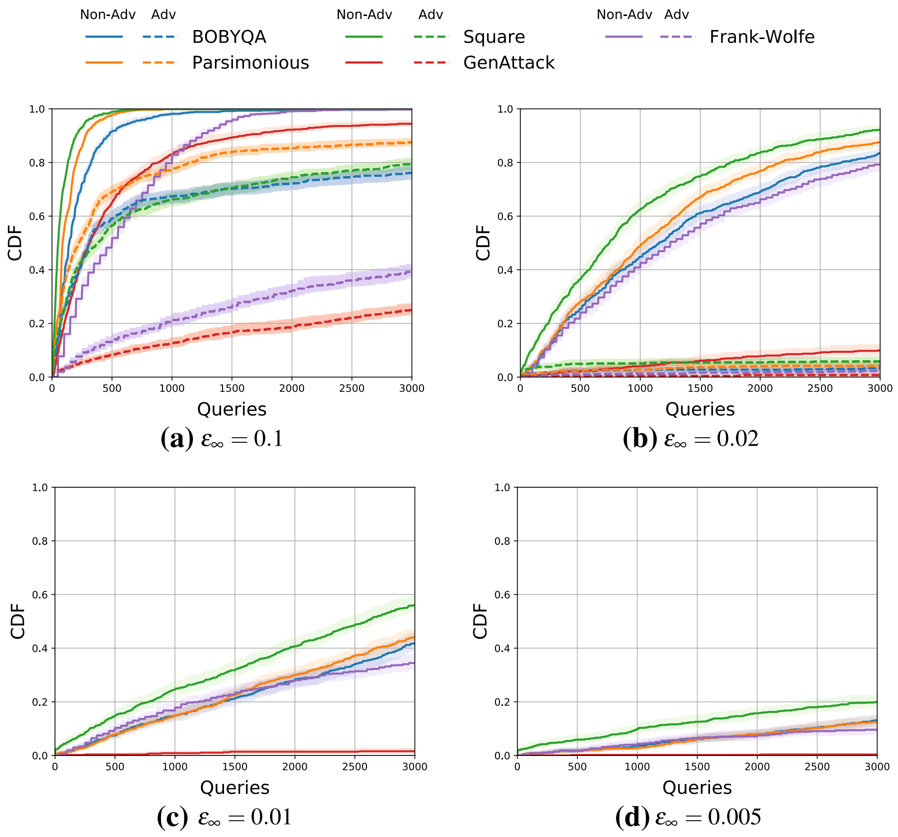


Fig. 10 Cumulative fraction of test set images successfully misclassified with adversarial examples generated by GenAttack, Parsimonious, Square, Frank-Wolfe, and our BOBYQA based approaches for different maximum perturbation energies ε_∞ and DNNs trained on the CIFAR10 dataset. In all results the solid and dashed lines denoted by 'Non-Adv' and 'Adv' corresponds to attacks on networks trained without or with the MadryLab defense strategy (Engstrom et al. 2019) respectively. The shaded region corresponds to the 90% confidence intervals for each method computed via boot-strapping the data

particular, we do this by sampling 100 times 50% of the data with reinsertion for both the CIFAR10 and ImageNet case. In Figure 10 and 11 the 90% confidence interval is plotted in the shaded regions while the median result is plotted with the continuous line.

In the CIFAR10 case, the confidence regions are very near to the median values suggesting that the results that have been plotted in the main text have a very small variance. Since the shaded regions are almost distinct, the comparison between the methods is almost sure. On the other hand, the results relative to ImageNet have a larger confidence interval due to the fewer attacks considered. This is especially relevant for low energies as the confidence intervals overlap, and thus the median CDF of a method being higher than the other implies that

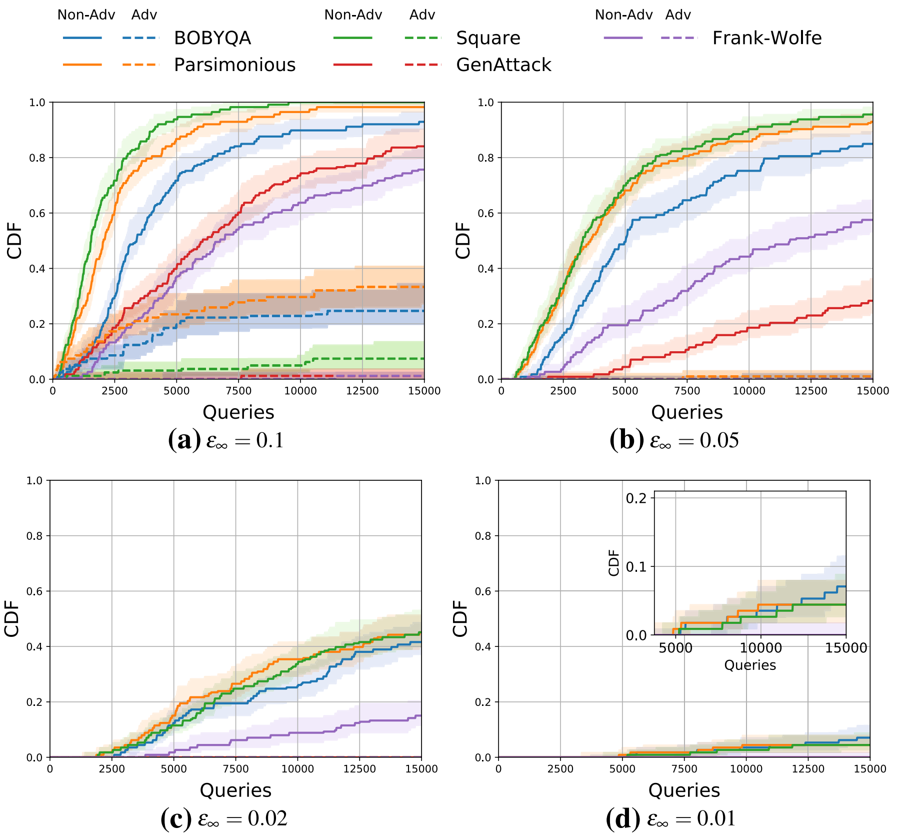


Fig. 11 Cumulative fraction of test set images successfully misclassified with adversarial examples generated by GenAttack, Parsimonious, Square, Frank-Wolfe, and our BOBYQA based approaches for different maximum perturbation energies ε_∞ and DNNs trained on the ImageNet dataset. In all results the solid and dashed lines denoted by 'Non-Adv' and 'Adv' corresponds to attacks on networks trained without or with the MadryLab defense strategy (Engstrom et al. 2019) respectively. The shaded region corresponds to the 90% confidence intervals for each method computed via boot-strapping the data

one is more efficient than the other only with some probability. For example, for $\varepsilon = 0.01$ the BOBYQA algorithm is better than square with a confidence of 70%.

References

Al-Dujaili A, O'Reilly UM (2020) There are no bit parts for sign bits in black-box attacks. In: Proceedings of the International Conference on Learning Representations (ICLR)

Alzantot M, Sharma, Y, Chakraborty S, Zhang H, Hsieh CJ, Srivastava MB (2019) Genattack: Practical black-box attacks with gradient-free optimization. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO), p. 1111–1119. https://doi.org/10.1145/3321707.3321749

Andriushchenko M, Croce F, Flammarion N, Hein M (2020) Square attack: a query-efficient black-box adversarial attack via random search. Proceedings of the European Conference on Computer Vision (ECCV)

Carlini N, Wagner D (2017) Towards evaluating the robustness of neural networks. In: Proceedings of the IEEE Symposium on Security and Privacy (SP), pp. 39–57. doi: https://doi.org/10.1109/SP.2017.49

Cartis C, Ferguson T, Roberts L (2020) Scalable derivative-free optimization for nonlinear least-square problems. Beyond First-Order Methods in ML Systems workshop at ICML

Cartis C, Fiala J, Marteau B, Roberts L (2019) Improving the flexibility and robustness of model-based derivative-free optimization solvers. ACM Trans Math Softw. https://doi.org/10.1145/3338517

Chen J, Zhou D, Yi J, Gu Q (2020) A frank-wolfe framework for efficient and effective adversarial attacks. In: Proceedings of the Association for the Advancement of Artificial Intelligence Conference (AAAI), pp. 3486–3494

Chen PY, Sharma Y, Zhang H, Yi J, Hsieh CJ (2018) Ead: elastic-net attacks to deep neural networks via adversarial examples. In: Proceedings of the AAAI Conference on Artificial Intelligence, pp. 10–17

Chen PY, Zhang H, Sharma Y, Yi J, Hsieh CJ (2017) ZOO: zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In: Proceedings of the ACM Workshop on Artificial Intelligence and Security (AISec), p. 15–26. https://doi.org/10.1145/3128572.3140448

Conn AR, Scheinberg K, Vicente LN (2009) Introduction to derivative-free optimization, vol 8. SIAM, Delhi

Dalvi N, Domingos P, Sanghai S, Verma D et al (2004) Adversarial classification. Proceedings of the ACM International conference on Knowledge Discovery and Data Mining (SIGKDD) 99–108. https://doi.org/10.1145/1014052.1014066

Dhillon GS, Azizzadenesheli K, Lipton ZC, Bernstein J, Kossaifi J, Khanna A, Anandkumar A (2018) Stochastic activation pruning for robust adversarial defense. In: Proceedings of the International Conference on Learning Representations (ICLR)

Efron B, Tibshirani RJ (1994) An introduction to the bootstrap. CRC Press, USA

Engstrom L, Ilyas A, Santurkar S, Tsipras D (2019) Robustness (python library). https://github.com/MadryLab/robustness

Eykholt K, Evtimov I, Fernandes E, Li B, Rahmati A, Xiao C, Prakash A, Kohno T, Song D (2018) Robust physical-world attacks on deep learning visual classification. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1625–1634. https://doi.org/10.1109/CVPR.2018.00175

Goodfellow IJ, Shlens J, Szegedy C (2015) Explaining and harnessing adversarial examples. In: Proceedings of the International Conference on Learning Representations (ICLR)

Gopalakrishnan S, Marzi Z, Madhow U, Pedarsani R (2018) Toward robust neural networks via sparsification. arXiv preprint arXiv:1810.10625

Gratton S, Sartenaer A, Toint PL (2008) Recursive trust-region methods for multiscale nonlinear optimization. SIAM J Optim 19(1):414–444

Guo C, Frank JS, Weinberger KQ (2018) Low frequency adversarial perturbation. In: Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)

Guo C, Rana M, Cisse M, van der Maaten L (2018) Countering adversarial images using input transformations. In: Proceedings of the International Conference on Learning Representations (ICLR). https://openreview.net/forum?id=SyJ7ClWCb

Hao-Chen H.X.Y.M, Deb L.D, Anil H.L.J.L.T, Jain K (2020) Adversarial attacks and defenses in images, graphs and text: a review. Int J Autom Comput 17(2):151–178

He K, Zhang X, Ren S, Sun J (2015) Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV), p. 1026–1034. https://doi.org/10.1109/ICCV.2015.123

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

Hinton G, Deng L, Yu D, Dahl GE, Mohamed A, Jaitly N, Senior A, Vanhoucke V, Nguyen P, Sainath TN, Kingsbury B (2012) Deep neural networks for acoustic modeling in speech recognition: the shared views of four research groups. IEEE Signal Process Mag 29(6):82–97. https://doi.org/10.1109/MSP.2012.2205597

Ilyas A, Engstrom L, Athalye A, Lin J (2018) Black-box adversarial attacks with limited queries and information. In: Proceedings of the International Conference on Machine Learning (ICML), pp. 2137–2146

Ilyas A, Engstrom L, Madry A (2019) Prior convictions: black-box adversarial attacks with bandits and priors. In: Proceedings of the International Conference on Learning Representations (ICLR)

Kurakin A, Goodfellow I Bengio S (2017) Adversarial examples in the physical world. In: Proceedings of the International Conference on Learning Representations (ICLR), Workshop Track

Larson J, Menickelly M, Wild SM (2019) Derivative-free optimization methods. Acta Numerica 28:287–404. https://doi.org/10.1017/S0962492919000060

Liu Y, Chen X, Liu C, Song D (2017) Delving into transferable adversarial examples and black-box attacks. Proceedings of the International Conference on Learning Representations (ICLR)

Monti F, Frasca F, Eynard D, Mannion D, Bronstein MM (2019) Fake news detection on social media using geometric deep learning. arXiv preprint arXiv:1902.06673

Moon S, An G, Song HO (2019) Parsimonious black-box adversarial attacks via efficient combinatorial optimization. In: Proceedings of the International Conference on Machine Learning (ICML), pp. 4636–4645

Moré JJ, Wild SM (2009) Benchmarking derivative-free optimization algorithms. SIAM J Optim 20(1):172–191

Narodytska N, Kasiviswanathan S (2017) Simple black-box adversarial attacks on deep neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), pp. 1310–1318 . https://doi.org/10.1109/CVPRW.2017.172

Naseer M, Khan S, Porikli F (2019) Local gradients smoothing: Defense against localized adversarial attacks. In: 2019 IEEE Winter Conference on Applications of Computer Vision (WACV), pp. 1300–1307

Nocedal J, Wright SJ (2006) Numerical optimization. Springer, New York. https://doi.org/10.1007/978-0-387-40065-5

Papernot N, McDaniel P, Goodfellow I, Jha S, Celik ZB, Swami A (2017) Practical black-box attacks against machine learning. In: Proceedings of the ACM on Asia Conference on Computer and Communications Security (ASIA CCS), p. 506–519 . https://doi.org/10.1145/3052973.3053009

Powell MJ (2009) The bobyqa algorithm for bound constrained optimization without derivatives. Tech. Rep. DAMTP 2009/NA06, University of Cambridge

Roberts L (2019) Derivative-free algorithms for nonlinear optimisation problems. Ph.D. thesis, University of Oxford

Scheinberg K, Toint P (2010) Self-correcting geometry in model-based algorithms for derivative-free unconstrained optimization. SIAM J. Optim. 20:3512–3532

Sharma Y, Ding GW, Brubaker M (2019) On the effectiveness of low frequency perturbations. Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI) pp. 3389–3396 https://doi.org/10.24963/ijcai.2019/470

Sitawarin C, Bhagoji AN, Mosenia A, Chiang M, Mittal P (2018) Darts: Deceiving autonomous cars with toxic signs. arXiv preprint arXiv:1802.06430

Szegedy C, Zaremba W, Sutskever I, Bruna J, Erhan D, Goodfellow I, Fergus R (2014) Intriguing properties of neural networks. In: Proceedings of the International Conference on Learning Representations (ICLR)

Tett SFB, Mineter MJ, Cartis C, Rowlands DJ, Liu P (2013) Can top-of-atmosphere radiation measurements constrain climate predictions? part i: Tuning. J Clim 26(23):9348–9366. https://doi.org/10.1175/JCLI-D-12-00595.1

Tu CC, Ting P, Chen PY, Liu S, Zhang H, Yi J, Hsieh CJ, Cheng SM (2019) Autozoom: Autoencoder-based zeroth order optimization method for attacking black-box neural networks. In: Proceedings of the AAAI Conference on Artificial Intelligence: Special Technical Track: AI for Social Impact. https://doi.org/10.1609/aaai.v33i01.3301742

Ughi G, Abrol V, Tanner J (2019) A model-based derivative-free approach to black-box adversarial esx-amples: Bobyqa. Proceedings of the Neural Information Processing Systems (NeruIPS) workshop "Beyond First Order Methods in ML"

Wang X, Wang S, Chen PY, Wang Y, Kulis B, Lin X, Chin S (2019) Protecting neural networks with hierarchical random switching: Towards better robustness-accuracy trade-off for stochastic defenses. In: Proceedings of the International Joint Conference on Artificial Intelligence, IJCAI

Yuan X, He P, Zhu Q, Li X (2019) Adversarial examples: attacks and defenses for deep learning. IEEE Trans Neural Netw Learn Syst 30(9):2805–2824. https://doi.org/10.1109/TNNLS.2018.2886017