Sparse and Imperceivable Adversarial Attacks

Francesco Croce
University of Tübingen

Matthias Hein
University of Tübingen

Abstract

Neural networks have been proven to be vulnerable to a variety of adversarial attacks. From a safety perspective, highly sparse adversarial attacks are particularly dangerous. On the other hand the pixelwise perturbations of sparse attacks are typically large and thus can be potentially detected. We propose a new black-box technique to craft adversarial examples aiming at minimizing l_0-distance to the original image. Extensive experiments show that our attack is better or competitive to the state of the art. Moreover, we can integrate additional bounds on the componentwise perturbation. Allowing pixels to change only in region of high variation and avoiding changes along axis-aligned edges makes our adversarial examples almost non-perceivable. Moreover, we adapt the Projected Gradient Descent attack to the l_0-norm integrating componentwise constraints. This allows us to do adversarial training to enhance the robustness of classifiers against sparse and imperceivable adversarial manipulations.

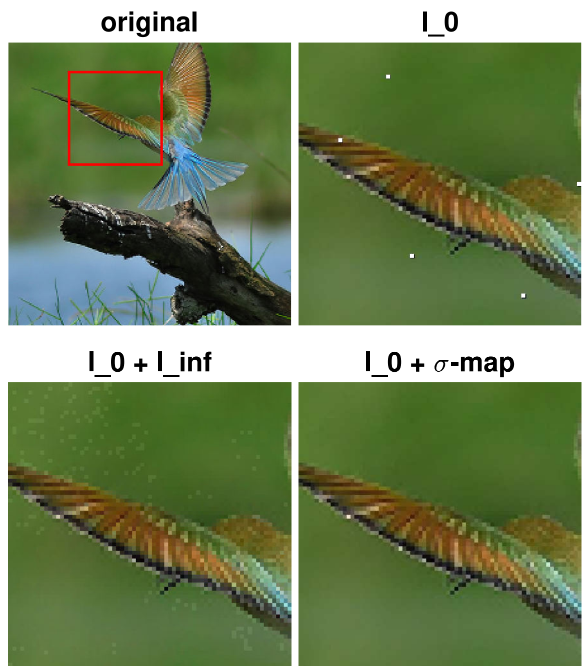

Figure 1: Upper left: original image with box for zoom, Upper Right: our l_0-attack, very few pixels, only 0.04%, are changed, but the modified pixels are clearly visible, Lower left: the result of the $l_0 + l_\infty$-attack as proposed in [22], the modifications are sparse, 2.7% of the pixels are changed, but clearly visible, Lower right: our sparse, 2.7% of the pixels are changed, but imperceivable attack ($l_0 + \sigma$-map).

1. Introduction

State-of-the-art neural networks are not robust [3, 30, 12], in the sense that a very small adversarial change of a correctly classified input leads to a wrong decision. While [30, 12] have brought up this problem in object recognition tasks, the problem itself has been discussed for some time in the area of email spam classification [9, 19]. This non-robust behavior of neural networks is a problem when such classifiers are used for decision making in safety-critical systems e.g. in autonomous driving or medical diagnosis systems. Thus it is important to be aware of the possible vulnerabilities as they can lead to fatal failures beyond the eminent security issue [18].

Recent research on attacks can be divided into white-box attacks [23, 6, 5, 21, 8], where one has access to the model at attack time, and black box attacks [7, 4, 13, 2] where one can just query the output of the classifier or the confidence scores of all classes. Typically the attacks try to find points on or close to the decision boundary, where the distance is measured in the pixels space, most often wrt the l_∞- and l_2-norm [23, 6, 5, 8], or one tries to maximize the loss resp. minimize the confidence in the correct class in some ϵ-ball around the original image [21]. Non pixelwise attacks exploiting geometric transformations have been proposed in [14, 32]. While it has been argued that adversarial changes will not happen in practical scenarios, this argument has been refuted in [16, 11]. Adversarial attacks during training have been early on proposed as a potential defense [30, 12], now known as adversarial training. In the form proposed in [21] this is one of the few defenses which could not be broken easily [5, 1].

In this paper we are dealing specifically with sparse adversarial attacks, that is we want to modify the smallest amount of pixels in order to change the decision. There are currently white-box attacks based on variants of gradient based methods integrating the l_0-constraint [6, 22] or mainly black-box attacks which use either local search or evolutionary algorithms [24, 29, 28]. The paper has the following methodological contributions: 1) we suggest a novel black-box attack based on local search which outperforms all existing l_0-attacks, 2) we present closed form expressions or simple algorithms for the projections onto the l_0-ball (or intersection of l_0-ball and componentwise constraints) in order to extend the PGD attack of [21] to the considered scenario, 3) since sparse attacks are often clearly visible and thus, at least in some cases, easy to detect (see upper right image of Figure 1), we combine the sparsity constraint (l_0-ball) with componentwise constraints, and we extend the two l_0-attacks mentioned above to produce sparse and imperceivable adversarial perturbations.

Compared to [22] who introduce global componentwise constraints (see lower left image of Figure 1) we propose to use locally adaptive componentwise constraints. These local constraints ensure that the change is typically not visible, that is we neither change color too much, nor we change pixels along edges aligned with the coordinate axis (see the Appendix for a visualization) or in regions which have uniform color (see lower right image of Figure 1). This is in line with, and significantly improves upon, [20], who suggest to perturb pixels in regions of high variance to have less recognizable modifications. In fact the often employed l_∞-attacks which modify each pixel only slightly but have to manipulate all pixels seem not to model perturbations which could actually occur. We think that our sparse and imperceivable attacks could happen in practice and correspond to modifications which do not change the semantics of the images even on very small scales. The good news of our paper is that the success rate of such attacks (50-70% success rate for standard models) is smaller than that of the commonly used ones - nevertheless we find it disturbing that such manipulations are possible at all. Thus we also test if adversarial training can reduce the success rate of such attacks. We find that adversarial training wrt l_2 partially decreases the effectiveness of l_0-attacks, while adversarial training wrt either l_2 or l_∞ helps to be more robust against sparse and imperceivable attacks. Finally, we introduce adversarial training aiming specifically at robustness wrt both our attack models.

2. Sparse and imperceivable adversarial attacks

Let $f : \mathbb{R}^d \longrightarrow \mathbb{R}^K$ be a multi-class classifier, where d is the input dimension and K the number of classes. A test point $x \in \mathbb{R}^d$ is classified as $c = \operatorname*{arg\,max}_{r=1,\ldots,K} f_r(x)$. The minimal adversarial perturbation y^* of $x \in \mathbb{R}^d$ with respect to a distance function $\gamma : \mathbb{R}^d \longrightarrow \mathbb{R}_+$ is given as the solution $y^* \in \mathbb{R}^d$ of the optimization problem

$$\begin{aligned} \min_{y\in\mathbb{R}^d} \quad & \gamma(y-x) \\ \text{s.th.} \quad & \operatorname*{arg\,max}_{r=1,\ldots,K} f_r(y) \neq \operatorname*{arg\,max}_{r=1,\ldots,K} f_r(x), \\ & y \in C, \end{aligned} \tag{1}$$

where C is a set of constraints valid inputs need to satisfy (e.g. images are scaled to be in $[0,1]^d$). Said otherwise: y^* is the closest point to x wrt the distance function γ which is classified differently from x.

2.1. Sparse l_0-attack

In an l_0-attack one is interested in finding the smallest number of pixels which need to be changed so that the decision changes. We write in the following gray-scale images x with d pixels as vectors in $[0,1]^d$ and color images x with d pixels as matrices x in $[0,1]^{d\times 3}$, and x_i denotes the i-th pixel with the three color channels in RGB. The corresponding distance function γ is thus given for gray-scale images as the standard l_0-norm

$$\gamma(y-x) = \sum_{i=1}^{d} \mathbb{1}_{|y_i - x_i| \neq 0}, \tag{2}$$

and for color images as

$$\gamma(y-x) = \sum_{i=1}^{d} \max_{j=1,\ldots,3} \mathbb{1}_{|y_{ij} - x_{ij}| \neq 0}, \tag{3}$$

where the inner maximization checks if any color channel j of the pixel i is changed. From a practical point of view the l_0-attack tests basically how vulnerable the model is to failure of pixels or large localized changes on an object e.g. a sticker on a refrigerator or dirt/dust on a windshield.

2.2. Sparse and Imperceivable attack

The problem of l_0-attacks is that they are completely unconstrained in the way how they change each pixel. Thus the perturbed pixels have usually completely different color than the surrounding ones and thus are easily visible. On the other hand l_∞-attacks, using the distance function

$$\gamma(y-x) = \max_{i=1,\ldots,d} \max_{j=1,\ldots,3} |y_{ij} - x_{ij}|,$$

are known to result in very small changes per pixel but have to modify every pixel and color channel. This seems to be a quite unrealistic perturbation model from a practical point of view. A much more realistic attack model which could happen in a practical scenario is when the changes are sparse but also imperceivable. In order to achieve this we come up with additional constraints on the allowed channelwise change. In [22] they suggest to have global bounds, for some fixed $\delta > 0$, in the form

$$x_{ij} - \delta \leq y_{ij} \leq x_{ij} + \delta,$$

which should ensure that the changes are not visible (we call an attack with l_0-norm and these global componentwise bounds an $l_0 + l_\infty$-attack in the following). However, these global bounds are completely agnostic of the image and thus δ has to be really small so that the changes are not visible even in regions of homogeneous color, e.g. sky, where almost any variation is easily spotted. We suggest image-specific local bounds taking into account the image structure. We have two specific goals:

1) We do not want to make changes along edges which are aligned with the coordinate axis as they can be easily spotted and detected.

2) We do not want to change the color too much and rather just adjust its intensity and keep approximately also its saturation level.

In order to achieve this we compute the standard deviation of each color channel in x- and y-axis directions with the two immediate neighboring pixels and the original pixel. We denote the corresponding values as $\sigma_{ij}^{(x)}$ and $\sigma_{ij}^{(y)}$ and define $\sigma_{ij} = \sqrt{\min\{\sigma_{ij}^{(x)}, \sigma_{ij}^{(y)}\}}$. Since $\sigma_{ij}^{(x)}, \sigma_{ij}^{(y)} \in [0,1]$ the square root increases more significantly, in relative value, smaller $\min\{\sigma_{ij}^{(x)}, \sigma_{ij}^{(y)}\}$. In this way we both enlarge the space of the possible adversarial examples and prevent perturbations in areas of zero variance. In fact we allow the changed image y just to have values given by

$$y_{ij} = (1 + \lambda_i \sigma_{ij}) x_{ij}, \quad \text{with } -\kappa \leq \lambda_i \leq \kappa, \tag{4}$$

where $\kappa > 0$. Additionally, we enforce box constraints $y \in [0,1]^{d \times 3}$. Note that the parameter λ_i corresponds to a change in intensity of pixel i by maximally plus/minus $\kappa \sum_{j=1}^{3} \sigma_{ij} x_{ij}$ as

$$\sum_{j=1}^{3} y_{ij} = \sum_{j=1}^{3} x_{ij} + \lambda_i \sum_{j=1}^{3} \sigma_{ij} x_{ij}.$$

Thus we are just changing intensity of the pixel instead of the actual color. Moreover, note that this change also preserves the saturation of the color value[1] if the σ_{ij} are equal for $j = 1, \ldots, 3$. Thus we fulfill the second requirement from above. Moreover, the first requirement is satisfied as $\sigma_{ij} = \sqrt{\min\{\sigma_{ij}^{(x)}, \sigma_{ij}^{(y)}\}}$, meaning that if along one of the coordinates there is no change in all color channels then the pixel cannot be modified at all. Thus pixels along a coordinate-aligned edge showing no change in color will not be changed. The attack model of sparse and imperceivable attacks will be abbreviated as $l_0 + \sigma$-map. For gray-scale images $x \in [0,1]^d$ we use instead

$$y_i = x_i + \lambda_i \sigma_i, \quad \text{with } -\kappa \leq \lambda_i \leq \kappa. \tag{5}$$

as there the approximate preservation of color saturation is not needed.

[1] In the HSV color space the saturation of a color is defined as $1 - \frac{\min\{R,G,B\}}{\max\{R,G,B\}}$, where R, G, B are the red/green/blue color channels in RGB color space.

3. Algorithms for sparse (and imperceivable) attacks

In this paper we propose two methods to generate l_0-, $l_0 + l_\infty$ and $l_0 + \sigma$-attacks. The first one is a randomized black-box attack based on the logits (the output of the neural network before the softmax layer) of the classifier. The second is a generalization of projected gradient descent (PGD) on the loss of the correct label [21] to our different attack models. For each attack model we will derive algorithms for the projection onto the corresponding sets.

3.1. Score-based sparse (and imperceivable) attack

Most of the existing black-box l_0-attacks either start with perturbing a small set of pixels and then enlarge this set until they find an adversarial example [26, 24] or, given a successful adversarial manipulation, try to progressively reduce the number of pixels exploited to change the classification [6, 28]. Instead we introduce a flexible attack scheme where at the beginning one checks pixelwise targeted attacks and then sorts them according to the resulting gap in the classifier outputs. Then we introduce a probability distribution on the sorted list and sample one-pixel changes to generate attacks where more pixels are manipulated simultaneously. The distribution we use is biased towards the one-pixel perturbations which produce, when applied individually, already large changes in the classifier output. In this non-iterative scheme there is thus no danger to get stuck in suboptimal points. Moreover, while the attack has to test many points, its non-iterative nature allows to check the perturbed points in large batches which is thus much faster than an evolutionary attack. Even if the scheme is simple it outperforms all existing methods including white-box attacks.

One-pixel modifications In the first step we check all one pixel modifications of the original image $x \in [0,1]^{d \times 3}$ (color) or $x \in [0,1]^d$ (gray-scale). The tested modifications depend on the attack model.

1. l_0**-attack:** for each pixel i we generate $8 = 2^3$ images changing the original color value to one of the 8 corners of the RGB color cube. Thus we name our method

CornerSearch. This results in a set of $8d$ images, all one pixel modifications of the original image x, which we denote by $(z^{(j)})_{j=1}^{8d}$. For gray-scale images one just checks the extreme gray-scale values (black and white) and gets $(z^{(j)})_{j=1}^{2d}$.

2. $l_0 + l_\infty$**-attack:** for each pixel i we generate 8 images changing the original color value of $(x_{ij})_{j=1}^{3}$ by the corners of the cube $[-\epsilon, \epsilon]^3$ resulting again in $(z^{(j)})_{j=1}^{8d}$ images. For gray-scale we use $x_i \pm \epsilon$ resulting in total in $(z^{(j)})_{j=1}^{2d}$ images. If necessary we clip to satisfy the constraint $z^{(j)} \in [0,1]^{d\times 3}$ or $z^{(j)} \in [0,1]^d$.

3. $l_0 + \sigma$**-map attack:** for color images we generate for each pixel i two images by setting

$$y_{ij} = (1 \pm \kappa\sigma_{ij})x_{ij}, \quad j = 1, \dots, 3,$$

where κ and σ_{ij} are as defined in Section 2. For gray scale images $x \in [0,1]^d$ we use

$$y_i = x_i \pm \kappa\sigma_i.$$

Finally, we clip y_{ij} and y_i to $[0,1]$. Thus, this results in $(z^{(j)})_{j=1}^{2d}$ images. We call it σ**-CornerSearch**.

After the generation of all the images we get the classifier output $f(z^{(j)})_{j=1}^{M}$ for each of them, where M is the total number of generated images, either $M = 2d$ or $M = 8d$. Then, separately for each class $r \neq c$, where $c = \operatorname*{arg\,max}_{r=1,\dots,K} f_r(x)$, we sort the values of

$$f_r(z^{(j)}) - f_c(z^{(j)})$$

in decreasing order $\pi^{(r)}$. That means for all $1 \le s \le M-1$

$$f_r(z^{(\pi_s^{(r)})}) - f_c(z^{(\pi_s^{(r)})}) \ge f_r(z^{(\pi_{s+1}^{(r)})}) - f_c(z^{(\pi_{s+1}^{(r)})}).$$

We introduce also an order $\pi^{(c)}$, sorting in decreasing order the quantities

$$\max_{r\neq c} f_r(z^{(j)}) - f_c(z^{(j)}).$$

The idea behind generating these one-pixel perturbations is to identify the pixels which push most the decision towards a particular class r or in case of the set $\pi^{(c)}$ towards an unspecific change. If $f_r(z^{(\pi_1^{(r)})}) - f_c(z^{(\pi_1^{(r)})}) > 0$ for some r, then the decision has changed by only modifying one pixel. In this case the algorithm stops immediately. Otherwise, one could try to iteratively select the most effective change and repeat the one-pixel perturbations. However, this is overly expensive and again suffers if suboptimal pixel modifications are chosen in the initial steps of the iterative scheme. Thus we suggest in the next paragraph a sampling scheme based on the obtained orderings, where one randomly selects k one-pixel modifications to combine in order to produce a multi-pixels attack.

Multi-pixels modifications Most of the times the modifications of one pixel are not sufficient to change the decision. Suppose we want to generate a candidate for a targeted adversarial sample towards class r by changing at most k pixels, choosing among the first N one-pixel perturbations according to the ordering $\pi^{(r)}$. We do this by sampling k indices $(s_1, \dots, s_k)$ in $\{1, \dots, N\}$ from the probability distribution on $\{1, \dots, N\}$ defined as

$$P(Z = i) = \frac{2N - 2i + 1}{N^2}, \quad i = 1, \dots, N. \tag{6}$$

The candidate image $y^{(r)}$ is generated by applying all the k one-pixel changes defined in the images $z^{(\pi_{s_1}^{(r)})}, \dots, z^{(\pi_{s_k}^{(r)})}$ to the original image x. Please note that we only sample from the top N one-pixel changes found in the previous paragraph and that the distribution on $\{1, \dots, N\}$ is biased towards sampling on the top of the list e.g. $P(Z = 1) = \frac{2N-1}{N^2}$ is $2N-1$ larger than $\mathrm{P}(Z = N) = \frac{1}{N^2}$. This bias ensures that we are mainly accumulating one-pixel changes which have led individually already to a larger change of the decision towards the target class r. We produce candidate images $y^{(1)}, \dots, y^{(K)}$ for all K classes, having $K-1$ candidate images targeted towards changes in a particular class and one image where the attack is untargeted (for $r = c$). In total we repeat this process N_{iter} times. The big advantage of the sampling scheme compared to an iterative scheme is that all these images can be fed into the classifier in batches in parallel which compared to a sequential processing is significantly faster. Moreover, it does not depend on previous steps and thus cannot get stuck in some suboptimal regions. As shown in the experiments this relatively simple sampling scheme performs better than sophisticated evolutionary algorithms (black-box attacks) and even white-box attacks.

Since we want to find adversarial examples differing from x in as few pixels as possible, we generate the batches $y^{(1)}, \dots, y^{(K)}$ of candidate images as described above, gradually increasing k, up to a threshold $k_{\max}$, until we get a classification different from the original class c. Algorithm 1 summarizes the main steps.

4. PGD for sparse and imperceivable attacks

The projected gradient descent (PGD) attack of Madry et al [21] is not aiming at finding the smallest adversarial perturbation but instead argues from the viewpoint of robust optimization about maximizing the loss

$$\max_{z\in C(x)} L(c, f(z)),$$

where $L : \{1, \dots, K\} \times \mathbb{R}^K \to \mathbb{R}_+$ is usually chosen to be the cross-entropy loss, c is the correct label of the point x and the set $C(x) \subset [0,1]^{d\times 3}$ (color images with d pixels) or $C(x) \subset [0,1]^d$ (gray-scale images). The interpretation

Algorithm 1: CornerSearch

Input : x original image classified as class c, K number of classes, N, $k_{\max}$, N_{iter}

Output: y adversarial example

$y \leftarrow \emptyset$
create *one-pixels modifications* $(z^{(i)})_{i=1}^{M}$
if *exists* $u \in (z^{(i)})_{i=1}^{M}$ *classified not as* c **then**
    $y \leftarrow u$, **return**
end
compute orderings $\pi^{(1)}, \dots, \pi^{(K)}$,
$k \leftarrow 2$
while $k \leq k_{max}$ **do**
    for $r = 1, \dots, K$ **do**
        create the set $Y^{(r)}$ of N_{iter} "k-pixels modifications" towards class r (see paragraph above)
        if $\exists u \in Y^{(r)}$ *classified not as* c **then**
            $y \leftarrow u$, **return**
        end
    end
    $k \leftarrow k + 1$
end

in terms of robust optimization [21] has led to a now well-accepted way of adversarial training with the goal of getting robust wrt a fixed set of perturbations. The usage of PGD attacks during training is the de facto standard for adversarial training, which we will also use later on in Section 5. Commonly used as the set of allowed perturbations is the l_∞-ball: $C(x) = \{z \,|\, \|z - x\|_\infty \leq \epsilon,\ z \in [0,1]^d\}$ as the projection can be done analytically.

In order to extend PGD to l_0- and l_0+l_∞-attacks, we first have to capture the sets allowed in our attack models in Section 2 and then find fast algorithms for the projections onto these sets. Once this is available PGD is ready to be used as an attack and for adversarial training. In the Appendix we also show how to project onto the intersection of the l_0-ball and the componentwise constraints given by the σ-map, for both color and gray-scale images. Thanks to this, we can introduce an $l_0 + \sigma$-map version of PGD, called σ-PGD, able to produce the sparse and imperceivable perturbations we have introduced.

4.1. Projection onto the l_0-ball and l_0+l_∞-ball

Given an original color image $x \in [0,1]^{d \times 3}$ we want to project a given point $y \in \mathbb{R}^{d \times 3}$ onto the set

$$C(x) = \Big\{z \in \mathbb{R}^{d\times 3} \,\Big|\, \sum_{i=1}^{d} \max_{j=1,2,3} \mathbb{1}_{|z_{ij}-x_{ij}|>0} \leq k,\ l_{ij} \leq z_{ij} \leq u_{ij}\Big\}.$$

We can write the projection problem onto $C(x)$ as

$$\begin{aligned}\min_{z \in \mathbb{R}^{d\times 3}} \quad & \sum_{i=1}^{d}\sum_{j=1}^{3} (y_{ij} - z_{ij})^2 \\ \text{s. th.} \quad & l_{ij} \leq z_{ij} \leq u_{ij}, \quad i = 1, \dots, d,\ j = 1, \dots, 3 \\ & \sum_{i=1}^{d} \max_{j=1,2,3} \mathbb{1}_{|z_{ij}-x_{ij}|>0} > 0 \leq k\end{aligned}$$

Ignoring the combinatorial constraint, we first solve for each pixel i the problem

$$\begin{aligned}\min_{z_i \in \mathbb{R}^{3}} \quad & \sum_{i=1}^{d}\sum_{j=1}^{3} (y_{ij} - z_{ij})^2 \\ \text{s. th.} \quad & l_{ij} \leq z_{ij} \leq u_{ij}, \quad i = 1, \dots, d,\ j = 1, \dots, 3\end{aligned}$$

The solution is given by $z_{ij}^* = \max\{l_{ij}, \min\{y_{ij}, u_{ij}\}\}$. We note that each pixel can be optimized independently from the other pixels. Thus we sort in decreasing order π the gains

$$\phi_i := \sum_{j=1}^{3} (y_{ij} - x_{ij})^2 - \sum_{j=1}^{3} (y_{ij} - z_{ij}^*)^2.$$

achieved by each pixel i. Thus the final solution differs from x in the k pixels (or less if there are less than k pixels with positive ϕ_i) which have the largest gain and is given by

$$z_{\pi_i j} = \begin{cases} z_{\pi_i j}^* & \text{for } i = 1, \dots, k,\ j = 1, \dots, 3, \\ x_{\pi_i j} & \text{else.} \end{cases}.$$

Using $l_{ij} = 0$ and $u_{ij} = 1$ we recover the projection onto the intersection of l_0-ball and $[0,1]^{d\times 3}$. For $l_0 + l_\infty$ note that the two constraints

$$0 \leq z_{ij} \leq 1, \quad -\epsilon \leq z_{ij} - x_{ij} \leq \epsilon,$$

are equivalent to:

$$\max\{0, -\epsilon + x_{ij}\} \leq z_{ij} \leq \min\{1, x_{ij} + \epsilon\}.$$

Thus by using

$$l_{ij} = \max\{0, -\epsilon + x_{ij}\}, \quad u_{ij} = \min\{1, x_{ij} + \epsilon\},$$

the set $C(x)$ is equal to the intersection of the l_0-ball of radius k, the l_∞-ball of radius ϵ around x and $[0,1]^{d\times 3}$,

5. Experiments

In the experimental section, we evaluate the effectiveness of our score-based l_0-attack CornerSearch and our white-box attack PGD_0. Moreover, we give illustrative examples of our sparse and imperceivable $l_0 + \sigma$-map attacks σ-CornerSearch and σ-PGD (the latter in the Appendix). Finally, we test adversarial training wrt various norms as a defense against our l_0- and $l_0 + \sigma$-map-attacks. The code is available at https://github.com/fra31/sparse-imperceivable-attacks.

		LocSearchAdv	PA 10x	CW	SparseFool	JSMA	CornerSearch
	black-box	Yes	Yes	No	No	No	Yes
MNIST	*success rate*	91.39%	92.35%	87.9%	100%	99.6%	97.38%
	mean (pixels)	17.56	8.82	46.04	19.44	83.92	9.21
	median (pixels)	-	8	44	12	46	7
CIFAR-10	*success rate*	97.32%	100%	100%	100%	100%	99.56%
	mean (pixels)	38.4	4.63	16.55	16.10	54.5	2.75
	median (pixels)	-	3	11	12	47	2

Table 1: **Comparison of different l_0-attacks.** While SparseFool is always successful it requires significantly more pixels to be changed. Our method CornerSearch requires out of all attacks the least *median* amount of pixels to be changed.

	SparseFool	CornerSearch
black-box	No	Yes
success rate	100%	93.26%
mean (pixels)	143.2	106.7
median (pixels)	101	50

Table 2: **l_0-attacks on Restricted ImageNet.** We attack the 89 correctly classified points out of 100 points from the validation set with SparseFool [22] and our algorithm CornerSearch. Due to the limit on the allowed number of pixel changes, CornerSearch is not always successful, but requires many less pixels to be changed.

5.1. Evaluation of l_0-attacks

We compare CornerSearch with state-of-the-art attacks for sparse adversarial perturbations: LocSearchAdv [24], Pointwise Attack (PA) [28], Carlini-Wagner l_0-attack (CW) [6], SparseFool (SF) [22], JSMA [26]. The first two operate in a black-box scenario, exploiting only the classifier output, like our method, while the latter three require access to the network itself (white-box attacks). Note that SparseFool is actually an l_1-attack, that means it uses the l_1-norm as distance measure in (1) in order to avoid the combinatorial problem arising from the usage of the l_0-norm. However, SparseFool can produce sparse attacks and in [22] has been shown to outperform l_0-attacks in terms of sparsity. We use the implementation of the Pointwise Attack in [27] with 10 restarts as done in [28], CW and JSMA from [25], while we reimplemented SparseFool. Since neither the code nor the models used in [24] are available (the results for LocSearchAdv are taken from [24]), we decided to compare the performance of the different attacks on one of the architectures reported in [24], the Network in Network [17] with batch normalization, retrained on MNIST and CIFAR-10.
We run the attacks on the first 1000 points of the corresponding test sets. We use CornerSearch with $k_{\max} = 50$, $N = 100$ and $N_{\text{iter}} = 1000$. In Table 1 we report the *success rate* of each method, that is the fraction of correctly

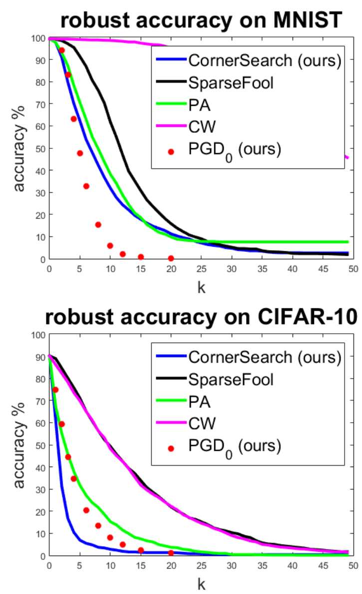

Figure 2: **Evaluation of PGD$_0$.** We compute for 1000 test points the robust accuracy of the classifier when the attack is allowed to perturb at most k pixels. We can see that PGD$_0$ (red dots) outperforms SparseFool, thus being the best "cheap" attack, and it is even the best one on MNIST for $k \geq 4$.

classified points which can be successfully attacked, *mean* and *median* number of pixels that every attack needs to modify to change the decision. Please recall that MNIST consists of images with 784 pixels and CIFAR-10 with 1024 pixels. Although CornerSearch does not find an adversarial example for each test point, since we fix the maximum number of pixels that can be modified, both the average and median number of changed pixels are lower than those of the other methods, that is less pixels need to be perturbed by our method to change the decision (with the only exception of the mean on MNIST, where anyway CornerSearch has higher success rate and lower median than PA). On MNIST CornerSearch requires for at least

50% of all test images 0.89% of the pixels to be changed and for CIFAR-10 it is even just 0.2%.

Using the derivation in Section 4 of the projection onto the l_0- resp. $l_0 + l_\infty$-ball, we introduce an l_0 version of the well-known PGD attack on the cross-entropy function L, namely PGD$_0$. The iterative scheme, to be repeated for a fixed number of iterations, is, given an input x assigned to class c,

$$\begin{aligned} z^{(i)} &= x^{(i-1)} + \eta \cdot {}^{\nabla L(c,f(x^{(i-1)})}/\|\nabla L(c,f(x^{(i-1)})\|_1 \\ x^{(i)} &= P_k(z^{(i)}), \end{aligned} \tag{7}$$

where $\eta \in \mathbb{R}_+$, $x^{(0)} = x$, $P_k(z)$ represents the projection onto the l_0-ball, with the radius fixed at k, and the l_∞-ball defined by the box constraint $x \in [0,1]^d$. Note that PGD$_0$ needs k to be specified and thus does not aim at the minimal modification to change the decision as in (1). In order to evaluate the robust accuracy, that is the accuracy of the classifier when the goal of the attacker is to change the decision of all correctly classified images using k-pixels modifications, one needs to evaluate PGD$_0$ for each value of k separately, whereas all other attacks yield the robust accuracy for all levels of sparsity in one run.
For comparison we run PGD$_0$, using 20 iterations and 10 random restarts, with 10 sparsity values k on the networks of Table 1 (see Appendix for more details). In Figure 2 we show the robust accuracy of the different attacks. PGD$_0$ achieves the best results on MNIST for $k \geq 4$, outperforms SparseFool and is even close to CornerSearch on CIFAR-10. As PGD$_0$ is very fast, it is a valuable alternative to our more expensive score-based attack.

We further test CornerSearch on Restricted ImageNet, that is a subset of ImageNet [10] where some of the classes are grouped to form 9 distinct macro-classes. We use the ResNet-50 from [31] and compare our attack to SparseFool [22] (we do not run the other methods as either no code is available or they do not scale to the size of the images). The images have 50176 pixels.
In Table 2 we report the statistics on 100 points for SparseFool and our attack with $k_{\max} = 1000, N_{\text{iter}} = 1000$. As for the other datasets, SparseFool always finds an adversarial example, whereas the smallest *mean* and *median* adversarial modification is achieved by CornerSearch, although with an inferior success rate. The runtime for SparseFool is around 55 times smaller than for CornerSearch. The runtime of our attack directly scales with the number of pixels and the time of a forward pass of the network, both large in this case. However, please note that SparseFool is a white-box attack, whereas ours is a black-box attack. For a comparison to PGD$_0$, given 100 pixels of budget, SF achieves a success rate of 49.4%, CS 64.0%, PGD$_0$ 39.3%.

	l_0				$l_0+\sigma$	
training	PA	SF	PGD$_0$	CS	σPGD	σCS
MNIST	$k=15$				$k=50$	
plain	25.6	41.2	**3.6**	9.2	49.2	80.4
l_∞-*at*	1.6	96.0	60.0	**0.0**	88.2	90.0
l_2-*at*	73.0	85.0	**34.0**	55.4	80.2	89.2
l_0-*at*	55.8	63.6	74.6	**39.8**	57.8	73.8
$l_0+\sigma$-*at*	19.4	26.8	**9.4**	15.4	93.6	95.4
CIFAR-10	$k=10$				$k=100$	
plain	15.2	57.0	7.0	**2.2**	27.6	52.4
l_∞-*at*	28.6	57.6	22.6	**10.8**	50.4	63.6
l_2-*at*	37.6	60.6	25.4	**13.8**	53.2	66.0
l_0-*at*	64.2	63.8	54.8	**46.0**	34.6	58.2
$l_0+\sigma$-*at*	41.6	54.4	6.0	**5.6**	62.8	67.6

Table 3: **Evaluation of adversarial training.** Robust accuracy (%) given by l_0- and $l_0+\sigma$-attacks (changing at most k pixels and for $l_0+\sigma$-attacks fixing $\kappa = 0.8$ for MNIST and $\kappa = 0.4$ for CIFAR-10) on models adversarially trained wrt different metrics.

5.2. Sparse and Imperceivable manipulations

We illustrate the differences of the adversarial modifications found by l_0-, $l_0 + l_\infty$- and $l_0+\sigma$-map attacks. In Figures 3 and 4 we show some examples. As discussed before the adversarial modifications produced wrt only the l_0-norm are the sparsest but also the easiest to recognize. The l_0+l_∞-attack provides images where, although the absolute value of the individual modification is bounded (we use here $\delta = 0.1$ for CIFAR-10, $\delta = 0.05$ for ImageNet), some perturbations are visible since either colors are not homogeneous with the neighbors (second rows of the left part of Figure 3 and right part of Figure 4) or modifications of an uniform background are introduced (second row of the right part of Figure 3 and left part of Figure 4). On the other hand, the adversarial modifications of σ-CornerSearch are imperceivable while still being very sparse (third rows of Figures 3 and 4), showing that the σ-map, also shown in the Figures rescaled so that the largest component is equal to 1, is able to correctly identify the area where a change is difficult to perceive (see in particular the zoomed images).
We provide a comparison of the adversarial examples crafted by σ-CornerSearch and σ-PGD in the Appendix.

5.3. Adversarial training

In order to increase robustness of the models to sparse adversarial manipulations, we adapt adversarial training to our cases. We use PGD$_0$ presented above for adversarial training in order to achieve robustness against l_0-attacks (l_0-*at*), while we use σ-PGD to enhance robustness against

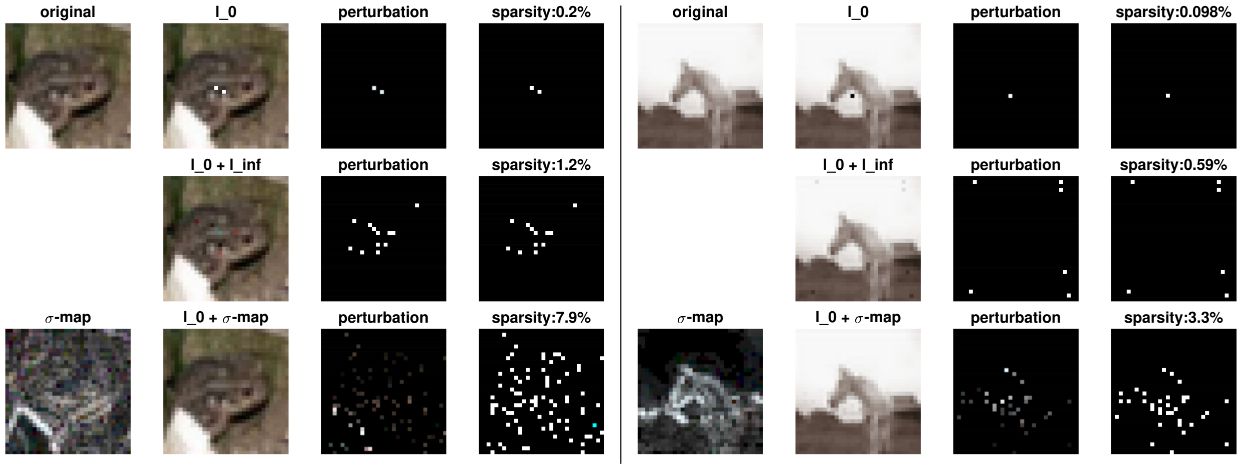

Figure 3: **Different attacks on CIFAR-10.** We illustrate the differences of the adversarial examples (second column) found by CornerSearch (l_0), $l_0 + l_\infty$-attack and σ-CornerSearch respectively first, second and third row. The third column shows the adversarial perturbations rescaled to $[0, 1]$, the fourth the map of the modified pixels (*sparsity* column). The original image can be found top left and the RGB representation of the σ-map bottom left.

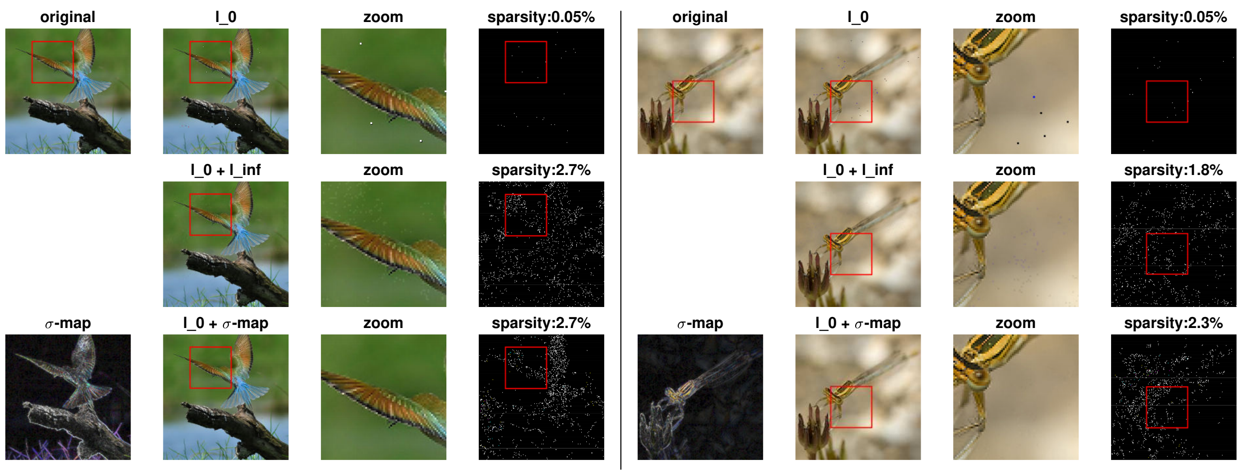

Figure 4: **Different attacks on Restricted ImageNet.** We illustrate the differences of the adversarial examples (second column, zoom in third column) found by CornerSearch (l_0), $l_0 + l_\infty$-attack and σ-CornerSearch, respectively first, second and third row. The fourth column shows the map of the modified pixels (*sparsity* column). The original image is in the top left and the RGB representation of the rescaled σ-map in the bottom left.

sparse and imperceivable attacks ($l_0 + \sigma$-*at*). With these two techniques we train models on MNIST and CIFAR-10 (more details about the architectures and hyerparameters in the Appendix). We compare them to the models trained on the *plain* training set and with adversarial training wrt the l_∞- and l_2-norm (l_∞-*at* and l_∞-*at*). In Table 3 we report the robust accuracy on 500 points (we fix the maximum number of pixels to be modified to k, and the parameter of the $l_0 + \sigma$ attacks defined in (4) and (5) to $\kappa = 0.8$ for MNIST and $\kappa = 0.4$ for CIFAR-10).

On MNIST the models trained on l_2 and l_0 perturbations are the most robust against l_0-attacks, while on CIFAR-10 the l_0-*at* model is more than 3 times more resistant than all the others. Similarly to [22] we find that l_∞-*at* does not help for l_0-robustness. Notably, on both dataset our attacks PGD_0 and CornerSearch (CS) achieve the best results and then are the most suitable to evaluate robustness.
Regarding the $l_0 + \sigma$-map attacks, we see that the $l_0 + \sigma$-*at* models are the least vulnerable, but also l_∞-*at* and l_2-*at* show some robustness. Note that σ-PGD is more successful than σ-CornerSearch but produces less sparse perturbations as it always fully exploits the budget of k pixels to modify while σ-CS mostly uses just a few of them, making the modifications even more difficult to spot (see Appendix).

References

[1] Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. *arXiv preprint arXiv:1802.00420*, 2018. 1

[2] Arjun Nitin Bhagoji, Warren He, Bo Li, and Dawn Xiaodong Song. Practical black-box attacks on deep neural networks using efficient query mechanisms. In *ECCV*, 2018. 1

[3] Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Srndic, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. Evasion attacks against machine learning at test time. In *ECML PKDD*, 2013. 1

[4] Wieland Brendel, Jonas Rauber, and Matthias Bethge. Decision-based adversarial attacks: Reliable attacks against black-box machine learning models. In *ICLR*, 2018. 1

[5] Nicholas Carlini and David A. Wagner. Adversarial examples are not easily detected: Bypassing ten detection methods. In *ACM Workshop on Artificial Intelligence and Security*, 2017. 1

[6] Nicholas Carlini and David A. Wagner. Towards evaluating the robustness of neural networks. In *IEEE Symposium on Security and Privacy*, 2017. 1, 2, 3, 6

[7] Pin-Yu Chen, Huan Zhang, Yash Sharma, Jinfeng Yi, and Cho-Jui Hsieh. 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)*, 2017. 1

[8] Francesco Croce and Matthias Hein. A randomized gradient-free attack on ReLU networks. In *GCPR*, 2018. 1

[9] Nilesh N. Dalvi, Pedro M. Domingos, Mausam, Sumit K. Sanghai, and Deepak Verma. Adversarial classification. In *KDD*, 2004. 1

[10] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kehui Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009. 7

[11] Ivan Evtimov, Kevin Eykholt, Earlence Fernandes, Tadayoshi Kohno, Bo Li, Atul Prakash, Amir Rahmati, and Dawn Song. Robust physical-world attacks on deep learning visual classification. In *CVPR*, 2018. 1

[12] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *ICLR*, 2015. 1

[13] Andrew Ilyas, Logan Engstrom, Anish Athalye, and Jessy Lin. Black-box adversarial attacks with limited queries and information. In *ICML*, 2018. 1

[14] Can Kanbak, Seyed-Mohsen Moosavi-Dezfooli, and Pascal Frossard. Geometric robustness of deep networks: analysis and improvement. In *CVPR*, 2018. 1

[15] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. preprint, arXiv:1412.6980, 2014. 12

[16] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. Adversarial examples in the physical world. In *ICLR Workshop*, 2017. 1

[17] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. In *ICLR*, 2014. 6, 11

[18] Y. Liu, X. Chen, C. Liu, and D. Song. Delving into transferable adversarial examples and black-box attacks. In *ICLR*, 2017. 1

[19] Daniel Lowd and Christopher Meek. Adversarial learning. In *KDD*, 2005. 1

[20] Bo Luo, Yannan Liu, Lingxiao Wei, and Qiang Xu. Towards imperceptible and robust adversarial example attacks against neural networks. In *AAAI*, 2018. 2

[21] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *ICLR*, 2018. 1, 2, 3, 4, 5, 11, 12

[22] Apostolos Modas, Seyed-Mohsen Moosavi-Dezfooli, and Pascal Frossard. Sparsefool: a few pixels make a big difference. In *CVPR*, 2019. 1, 2, 3, 6, 7, 8

[23] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *CVPR*, pages 2574–2582, 2016. 1

[24] Nina Narodytska and Shiva Prasad Kasiviswanathan. Simple black-box adversarial perturbations for deep networks. In *CVPR 2017 Workshops*, 2016. 2, 3, 6

[25] Nicolas Papernot, Fartash Faghri, Nicholas Carlini, Ian Goodfellow, Reuben Feinman, Alexey Kurakin, Cihang Xie, Yash Sharma, Tom Brown, Aurko Roy, Alexander Matyasko, Vahid Behzadan, Karen Hambardzumyan, Zhishuai Zhang, Yi-Lin Juang, Zhi Li, Ryan Sheatsley, Abhibhav Garg, Jonathan Uesato, Willi Gierke, Yinpeng Dong, David Berthelot, Paul Hendricks, Jonas Rauber, and Rujun Long. cleverhans v2.0.0: an adversarial machine learning library. preprint, arXiv:1610.00768, 2017. 6

[26] N. Papernot, P. D. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami. The limitations of deep learning in adversarial settings. In *1st IEEE European Symposium on Security & Privacy*, 2016. 3, 6

[27] Jonas Rauber, Wieland Brendel, and Matthias Bethge. Foolbox: A python toolbox to benchmark the robustness of machine learning models. In *ICML Reliable Machine Learning in the Wild Workshop*, 2017. 6, 11

[28] Lukas Schott, Jonas Rauber, Wieland Brendel, and Matthias Bethge. Towards the first adversarially robust neural network model on MNIST. In *ICLR*, 2019. 2, 3, 6

[29] Jiawei Su, Danilo Vasconcellos Vargas, and Kouichi Sakurai. One pixel attack for fooling deep neural networks. *arXiv preprint arXiv:1710.08864v5*, 2019. 2

[30] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. In *ICLR*, pages 2503–2511, 2014. 1

[31] Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. Robustness may be at odds with accuracy. In *ICLR*, 2019. 7, 11, 12

[32] Chaowei Xiao, Jun-Yan Zhu, Bo Li, Warren He, Mingyan Liu, and Dawn Song. Spatially transformed adversarial examples. In *ICLR*, 2018. 1