Smooth adversarial examples
Hanwei Zhang, Yannis Avrithis, Teddy Furon, Laurent Amsaleg

To cite this version:
Hanwei Zhang, Yannis Avrithis, Teddy Furon, Laurent Amsaleg. Smooth adversarial examples. EURASIP Journal on Information Security, 2020, 2020 (1), 10.1186/s13635-020-00112-z. hal-03017171

HAL Id: hal-03017171
https://hal.science/hal-03017171
Submitted on 8 Dec 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Smooth adversarial examples

Hanwei Zhang\textsuperscript{1,2}*, Yannis Avrithis\textsuperscript{1}, Teddy Furon\textsuperscript{1} and Laurent Amsaleg\textsuperscript{1}

Abstract

This paper investigates the visual quality of the adversarial examples. Recent papers propose to smooth the perturbations to get rid of high frequency artifacts. In this work, smoothing has a different meaning as it perceptually shapes the perturbation according to the visual content of the image to be attacked. The perturbation becomes locally smooth on the flat areas of the input image, but it may be noisy on its textured areas and sharp across its edges. This operation relies on Laplacian smoothing, well-known in graph signal processing, which we integrate in the attack pipeline. We benchmark several attacks with and without smoothing under a white box scenario and evaluate their transferability. Despite the additional constraint of smoothness, our attack has the same probability of success at lower distortion.

Keywords: Adversarial example, Image classification, Deep neural network

1 Introduction

Adversarial examples, where introduced by Szegedy et al.\textsuperscript{[1]} as imperceptible perturbations of a test image that can change a neural network’s prediction. This has spawned active research on adversarial attacks and defenses with competitions among research teams\textsuperscript{[2]}. Despite the theoretical and practical progress in understanding, the sensitivity of neural networks to their input, assessing the imperceptibility of adversarial attacks remains elusive: user studies show that $L_p$ norms are largely unsuitable, whereas more sophisticated measures are limited too\textsuperscript{[3]}. Machine assessment of perceptual similarity between two images (the input image and its adversarial example) is arguably as difficult as the original classification task, while human assessment of whether one image is adversarial is hard when the $L_p$ norm of the perturbation is small. Of course, when both images are available and the perturbation is isolated, one can always see it. To make the problem interesting, we ask the following question: given a single image, can the effect of a perturbation be magnified to the extent that it becomes visible and a human may decide whether this example is benign or adversarial?

Figure 1 shows that the answer is positive for a range of popular adversarial attacks. In Appendix 1, we propose a simple adversarial magnification producing a “magnified” version of a given image, without the knowledge of any other reference image. Assuming that natural images are locally smooth, this can reveal not only the existence of an adversarial perturbation but also its pattern. One can recognize, for instance, the pattern of Fig. 4 of\textsuperscript{[4]} in our Fig. 1f, revealing a universal adversarial perturbation.

Motivated by this example, we argue that popular adversarial attacks have a fundamental limitation in terms of imperceptibility that we attempt to overcome by introducing smooth adversarial examples. Our attack assumes local smoothness and generates examples that are consistent with the precise smoothness pattern of the input image. More than just looking “natural”\textsuperscript{[5]} or being smooth\textsuperscript{[6, 7]}, our adversarial examples are photorealistic, low-distortion, and virtually invisible even under magnification. This is evident by comparing our magnified example in Fig. 1d to the magnified original in Fig. 1b.

Given that our adversarial examples are more constrained, an interesting question is whether they perform well according to metrics like probability of success and $L_p$ distortion. We show that our attack not only is...
competitive but outperforms Carlini & Wagner [8], from which our own attack differs basically by a smoothness penalty.

1.1 Contributions
As primary contributions, we

1. Investigate the behavior of existing attacks when perturbations become “smooth like” the input image; and
2. Devise one attack that performs well on standard metrics while satisfying the new constraint.

As secondary contributions, we

3. Magnify perturbations to facilitate qualitative evaluation of their imperceptibility;
4. Show that properly integrating the smoothness constraint is not as easy as smoothing the perturbation generated by an attack; and
5. Define a new, more complete/fair evaluation protocol.

The remaining text is organized as follows. Section 2 formulates the problem and introduces a classification of attacks. It describes the C&W attack and the related work. Section 3 explains Laplacian smoothing, on which we build our method. Section 4 presents our **smooth**
adversarial attacks, and Section 5 provides experimental evaluation. Conclusions are drawn in Section 6. Our adversarial magnification used to generate Fig. 1 is specified in Appendix 1.

2 Problem formulation and related work

Let us denote by \( x \in \mathcal{X} := [0, 1]^{n \times d} \) an image of \( n \) pixels and \( d \) color channels that has been flattened in a given ordering of the spatial components. A classifier network \( f \) maps that input image \( x \) to an output \( y = f(x) \in \mathbb{R}^k \) which contains the logits of \( k \) classes. It is typically followed by softmax and cross-entropy loss at supervised training or contains the classification loss \( \ell \) (3) vanishes and the distortion term pulls \( \sigma(w) \) back towards \( x_o \), causing oscillations around the margin. Among all successful iterates, the one with the least distortion is kept; if there is none, the attack fails. The process is repeated for different Lagrangian multiplier \( \lambda \) according to line search. This family of attacks is typically more expensive than the two first.

2.2 Imperceptibility of adversarial perturbations

Adversarial perturbations are often invisible only because their amplitude is extremely small. Few papers deal with the need of improving the imperceptibility of the adversarial perturbations. The main idea in this direction is to create low or mid-frequency perturbation patterns. Zhou et al. [14] add a regularization term for the sake of transferability, which removes the high frequencies of the perturbation via low-pass spatial filtering. Heng et al. [6] propose a harmonic adversarial attack where perturbations are very smooth gradient-like images. Guo et al. [7] design an attack explicitly in the Fourier domain. However, in all cases above, the convolution and the bases of the harmonic functions and of the Fourier transform are independent of the visual content of the input image.

In contrast, the adversarial examples in this work are crafted to be locally compliant with the smoothness of the original image. Our perturbation may be sharp across the edges of \( x_o \) but smooth wherever \( x_o \) is, e.g., on background regions. It is not just smooth but photorealistic,

Szegedy et al. [1] approximate this constrained minimization problem by a Lagrangian formulation

\[
\min_{x \in \mathcal{X}} \|x - x_o\| + \lambda \ell(f(x), t). 
\]

Parameter \( \lambda \) controls the trade-off between the distortion and the classification loss. Szegedy et al. [1] carry out this optimization by box-constrained L-BFGS.

The attack of Carlini & Wagner [8], denoted C&W in the sequel, pertains to this approach. A change of variable eliminates the box constraint: \( x \in \mathcal{X} \) is replaced by \( \sigma(w) \), where \( w \in \mathbb{R}^{n \times d} \) is a latent vector and \( \sigma \) is the element-wise sigmoid function that projects \( \mathbb{R}^{n \times d} \) to \( \mathcal{X} \). A margin is introduced: an untargeted attack makes the logit \( y_i \) less than any other logit \( y_i \) for \( i \neq t \) by at least a margin \( m \geq 0 \). Similar to the multi-class SVM loss by Crammer and Singer [12] (where \( m = 1 \)), the loss function \( \ell \) is then defined as

\[
\ell(y, t) := [y_t - \max_{i \neq t} y_i + m]^+, 
\]

where \([ \cdot ]^+\) denotes the positive part. The C&W attack uses the Adam optimizer [13] to minimize the functional

\[
J(w, \lambda) := \lambda \|\sigma(w) - x_o\|^2 + \ell(f(\sigma(w)), t), 
\]

initializing by \( w_0 := \sigma^{-1}(x_o) \). When the margin is reached, loss \( \ell(y, t) \) vanishes and the distortion term pulls \( \sigma(w) \) back towards \( x_o \), causing oscillations around the margin. Among all successful iterates, the one with the least distortion is kept; if there is none, the attack fails. The process is repeated for different Lagrangian multiplier \( \lambda \) according to line search. This family of attacks is typically more expensive than the two first.

2.1 Families of attacks

In a white box setting, attacks typically rely on exploiting the gradient of some loss function. We propose to classify known attacks into three families.

2.1.1 Target distortion

This family gathers attacks targeting a distortion \( \epsilon \) given as an input parameter. Examples are early attacks like Fast Gradient Sign Method (FGSM) [9] and Iterative-FGSM (I-FGSM) [10]. Their performance is then measured by the probability of success \( P_{\text{succ}} := \mathbb{P}(p(x_o) \neq t) \) as a function of \( \epsilon \).

2.1.2 Target success

This family gathers attacks that always succeed in misclassifying \( x_o \), at the price of a possible large distortion. DeepFool [11] is a typical example. Their performance is then measured by the expected distortion \( \mathbb{D} := \mathbb{E}(\|x_o - x_o\|) \).

These two first families are implemented with variations of a gradient descent method. A classification loss function is defined on an output logit vector \( y = f(x) \) with respect to the original true label \( t \), denoted by \( \ell(y, t) \).

2.1.3 Target optimality

The above attacks are not optimal because they a priori do not solve the problem of succeeding under minimal distortion,

\[
\min_{x \in \mathcal{X}} \|x - x_o\|.
\]
because its smoothness pattern is guided by the input image. An analogy becomes evident with digital watermarking [15]. In this application, the watermark signal pushes the input image into the detection region (the set of images deemed as watermarked by the detector), whereas here the adversarial perturbation drives the image outside its class region. The watermark is invisible thanks to the masking property of the input image [16]. Its textured areas and its contours can hide a lot of watermarking power, but the flat areas cannot be modified without producing noticeable artifacts. Perceptually shaping the watermark signal allows a stronger power, which in turn yields more robustness.

Another related problem, with similar solutions mathematically, is photorealistic style transfer. Luan et al. [17] transfer style from a reference style image to an input image, while constraining the output to being photorealistic with respect to the input. This work as well as follow-up works [18, 19] is based on variants of Laplacian smoothing or regularization much like we do.

It is important to highlight that high frequencies can be powerful for deluding a network, as illustrated by the extreme example of the one pixel attack [20]. However, this is arguably one of the most visible attacks.

3 Background on graph Laplacian smoothing

Popular attacks typically produce noisy patterns that are not found in natural images. They may not be visible at first sight because of their low amplitude, but they are easily detected once magnified (see Fig. 1). Our objective is to craft an adversarial perturbation that is locally as smooth as the input image, remaining invisible through magnification. This section gives background on Laplacian smoothing [21, 22], a classical operator in graph signal processing [23, 24], which we adapt to images here. Section 4 uses it to generate a smooth perturbation guided by the original input image.

3.1 Graph

Laplacian smoothing builds on a weighted undirected graph whose n vertices correspond to the n pixels of the input image x. The i-th vertex of the graph is associated with feature x_i ≠ [0, 1]^d that is the i-th row of x, that is, x_i = [x_1, ..., x_n]^T. Matrix p = [p_1, ..., p_n]^T is the feature kernel, k is a spatial kernel, both being usually Gaussian or Laplacian. The spatial kernel is typically nonzero only on nearest neighbors, resulting in a sparse matrix W. We further define the n × n degree matrix D = diag(W) where I_n is the all-ones n-vector.

3.2 Regularization [21]

Now, given a new signal z ∈ RD with respect to the input. This work as well as the input image, remaining invisible through magnification, or identifies queries (for ranking), and is null for the remaining vertices. Smoothing then spreads the output signal s_a(z) := arg min r∈RD φ_a(r, z), with

\[ \phi_a(r, z) := \alpha \sum_{i,j} w_{ij} \| \hat{r}_i - \hat{r}_j \|^2 + (1 - \alpha) \| r - z \|^2_F \] (6)

where \( \hat{r} := D^{-1/2} r \) and \( \| \cdot \|_F \) is the Frobenius norm. The first summand is the smoothness term. It encourages \( \hat{r}_i \) to be close to \( \hat{r}_j \) when \( w_{ij} \) is large, i.e., when pixels i and j of input \( x_o \) are neighbors and similar. This encourages r to be smooth wherever \( x_o \) is. The second summand is the fitness term that encourages r to stay close to z. Parameter \( \alpha \in [0, 1) \) controls the trade-off between the two.

3.3 Filtering

If we symmetrically normalize matrix W as \( W := D^{-1/2} W D^{-1/2} \) and define the n × n regularized Laplacian matrix \( L_a := (I_n - \alpha W)/(1 - \alpha) \), then the expression (6) simplifies to the following quadratic form:

\[ \phi_a(r, z) := (1 - \alpha) \operatorname{tr}(r^T L_a r - 2z^T r + z^T z) . \] (7)

This reveals, by letting the derivative \( \partial \phi / \partial r \) vanish independently per column, that the smoothed signal is given in closed form:

\[ s_a(z) = L_a^{-1} z^* . \] (8)

This solution is unique because matrix \( L_a \) is positive-definite. Parameter \( \alpha \) controls the bandwidth of the smoothing; function \( s_a \) is the all-pass filter for \( \alpha = 0 \) and becomes a strict "low-pass" filter when \( \alpha \to 1 \) [25].

Variants of the model above have been used for instance for interactive image segmentation [22, 26, 27], transductive semi-supervised classification [21, 28], and ranking on manifolds [29, 30]. Input z expresses labels known for some input pixels (for segmentation) or samples (for classification), or identifies queries (for ranking), and is null for the remaining vertices. Smoothing then spreads the labels to these vertices according the weights of the graph.

3.4 Normalization

Contrary to applications like interactive segmentation or semi-supervised classification [21, 22], z does not represent a binary labeling but rather an arbitrary perturbation...
in this work. Also contrary to such applications, the output is neither normalized nor taken as the maximum over feature dimensions (channels). If \( L^{-1}_a \) is seen as a spatial filter, we therefore row-wise normalize it to one in order to preserve the dynamic range of \( z \). We therefore define the \textit{normalized smoothing function} as

\[
\hat{s}_a(z) := \text{diag}(s_a(1_a))^{-1}s_a(z).
\]

This function of course depends on \( x_o \). We omit this from notation but we say \( \hat{s}_a \) is \textit{smoothing guided} by \( x_o \) and the output is \textit{smooth like} \( x_o \).

4 Integrating smoothness into the attack

The key idea of the paper is that the smoothness of the perturbation is now consistent with the smoothness of the original input image \( x_o \), which is achieved by smoothing operations guided by \( x_o \). This section integrates smoothness into attacks targeting distortion (Section 4.1) and attacks targeting optimality (Section 4.2), but in very different ways.

4.1 Simple attacks

We consider here simple attacks targeting distortion or success based on gradient descent of the loss function. There are many variations which normalize or clip the update according to the norm used for measuring the distortion, a learning rate or a fixed step etc. These variants are loosely prototyped as the iterative process

\[
g = \nabla_x \ell \left( f(x^{(k)}_o), t \right),
\]

\[
x^{(k+1)}_o = c \left( x^{(k)}_o - n(g) \right),
\]

where \( c \) is a \textit{clipping} function and \( n \) a \textit{normalization} function according to the variant. Function \( c \) should at least produce a valid image: \( c(x) \in \mathcal{X} = [0,1]^{n \times d} \).

4.1.1 Quick and dirty

To keep these simple attacks simple, smoothness is loosely integrated after the gradient computation and before the update normalization:

\[
x^{(k+1)}_o = c \left( x^{(k)}_o - n(\hat{s}_o(g)) \right).
\]

This approach can be seen as a projected gradient descent on the manifold of perturbations that are smooth like \( x_o \). When applied to PGD, we call this attack qPGD where the “q” stands for a “quick and dirty” integration of the smoothness constraint.

4.2 Attack targeting optimality

This section integrates smoothness in the attacks targeting optimality like C&W. Our starting point is the unconstrained problem (4) [8]. However, instead of representing the perturbation signal \( r := x - x_o \) implicitly as a function \( \sigma(w) - x_o \) of another parameter \( w \), we express the objective explicitly as a function of variable \( r \), as in the original formulation of (2) in [1]. We make this choice because we need to directly process the perturbation \( r \). On the other hand, we now need the element-wise clipping function \( c(x) := \min(|x|_+,1) \) to satisfy the constraint \( x = x_o + r \in \mathcal{X} \).

\[
\min_r \lambda \|r\|^2 + \ell(f(c(x_o + r), t),
\]

where \( r \) is unconstrained in \( \mathbb{R}^{n \times d} \).

4.2.1 Smoothness penalty

At this point, optimizing (13) results in “independent” updates at each pixel. We would rather like to take the smoothness structure of the input \( x_o \) into account and impose a similar structure on \( r \). Representing the pairwise relations by a graph as discussed in Section 3, a straightforward choice is to introduce a pairwise loss term

\[
\mu \sum_{i,j} W_{ij} \| \hat{r}_i - \hat{r}_j \|^2
\]

into (13), where we recall that \( W_{ij} \) are the elements of the adjacency matrix \( W \) of \( x_o \). \( \hat{r}_i \) and \( D := \text{diag}(W1_o) \). A problem is that the spatial kernel on parameter \( r \) is globally smooth solution at each iteration.

4.2.2 Smoothness constraint

What we advocate instead is to apply a global smoothing process at each iteration: we introduce a \textit{latent variable} \( z \in \mathbb{R}^{n \times d} \) and seek for a joint solution with respect to \( r \) and \( z \) of the following

\[
\min_{r,z} \mu \phi_o(r, z) + \lambda \|r\|^2 + \ell(f(c(x_o + r), t),
\]

where \( \phi \) is defined by (6). In words, \( z \) represents an unconstrained perturbation, while \( r \) should be close to \( z \), smooth like \( x_o \), small, and such that the perturbed input \( x_o + r \) satisfies the classification objective. Then, by letting \( \mu \to \infty \), the first term becomes a hard constraint imposing a globally smooth solution at each iteration:

\[
\min_{r,z} \lambda \|r\|^2 + \ell(f(c(x_o + r), t)
\]

subject to \( r = \hat{s}_o(z) \),

where \( \hat{s}_o \) is defined by (9). During optimization, every iterate of this perturbation \( r \) is smooth like \( x_o \).
4.2.3 Optimization

With this definition in place, we solve for \( z \) the following unconstrained problem over \( \mathbb{R}^{n \times d} \):

\[
\min_z \lambda \left\| \hat{s}_v(z) \right\|^2 + \ell(f(c(x_o + \hat{s}_v(z))), t).
\]  

(18)

Observe that this problem has the same form as (13), where \( r \) has been replaced by \( \hat{s}_v(z) \). This implies that we can use the same optimization method as the C&W attack. The only difference is that the variable is \( z \), which we initialize by \( z = 0 \times d \), and we apply function \( \hat{s}_v \) at each iteration.

Gradients are easy to compute because our smoothing is a linear operator. We denote the loss on this new variable by \( L(z) := \ell(f(c(x_o + \hat{s}_v(z))), t) \). Its gradient is

\[
\nabla_L L(z) = J_{\hat{s}_v}(z)^\top \cdot \nabla_x \ell(f(c(x_o + \hat{s}_v(z))), t),
\]

(19)

where \( J_{\hat{s}_v}(z) \) is the \( n \times n \) Jacobian matrix of the smoothing operator at \( z \). Since our smoothing operator is defined by (8) and (9) is linear, \( J_{\hat{s}_v}(z) = \text{diag}(s_v(1_n))^{-1} L_v^{-1} \) is a matrix constant in \( z \), and multiplication by this matrix is equivalent to smoothing. The same holds for the distortion penalty \( \left\| \hat{s}_v(z) \right\|^2 \). This means that in the backward pass, the gradient of the objective (18) w.r.t. \( z \) is obtained from the gradient w.r.t. \( r \) (or \( x \)) by smoothing, much like how \( r \) is obtained from \( z \) in the forward pass (17).

Matrix \( L_v \) is fixed during optimization, depending only on input \( x_o \). For small images like in the MNIST dataset [31], it can be inverted: function \( \hat{s}_v \) is really a matrix multiplication. For larger images, we use the conjugate gradient (CG) method [32] to solve the set of linear systems \( L_v r = z \) for \( r \) given \( z \). Again, this is possible because matrix \( L_v \) is positive-definite, and indeed, it is the most common solution in similar problems [26,30,33]. At each iteration, one computes a product of the form \( v \mapsto L_v v \), which is efficient because \( L_v \) is sparse. In the backward pass, one can either use CG on the gradient, or auto-differentiate (AD) through the forward CG iterations. We choose the latter because it is the simplest implementation-wise. The two options have the same complexity and should have the same run-time in theory. In practice, Tensorflow AD takes 0.43 s on average for 50 CG iterations on ImageNet and InceptionV3, while CG forward takes 0.33 s.

4.2.4 Discussion

The clipping function \( c \) that we use is just the identity over the interval \([0,1]\), but outside this interval, its derivative is zero. Carlini & Wagner [8] therefore argue that the numerical solver of problem (13) suffers from getting stuck in flat spots: when a pixel of the perturbed input \( x_o + r \) falls outside \([0,1]\), it keeps having zero derivative after that and with no chance of returning to \([0,1]\) even if this is beneficial. This limitation does not apply to our case thanks to the \( L2 \) distortion penalty in (13) and to the updates in its neighborhood: such a value may return to \([0,1]\) thanks to the smoothing operation.

5 Experiments

Our experiments focus on the white box setting, where the defender first exhibits a network, and then the attacker mounts an attack specific to this network, but we also investigate a transferability scenario. All attacks are untargeted, as defined by loss function (3).

5.1 Evaluation protocol

For the perceptual evaluation of the quality of the adversarial images, we follow the recommendation of [34]. This paper compares fifteen metrics (including SSIM, PSNR, and wPSNR) to the subjective perceptual evaluation of a panel of users. The conclusion is that most apparent distortion (MAD) [35] is the metric best reflecting user assessment. A low MAD score means better fidelity.

For quantitative evaluation of the strength of an attack, we use two global statistics and an operating characteristic curve. Given a test image set of \( N \) images, we only consider its subset \( X \) of \( N \) images that are classified correctly without any attack. The accuracy of the classifier is \( N/N' \). Let \( X_{\text{succ}} \) be the subset of \( X \) with \( N_{\text{succ}} := |X_{\text{succ}}| \) where the attack succeeds and let \( D(x_o) := \|x_o - x_o\| \) be the distortion for image \( x_o \in X_{\text{succ}} \).

The global statistics are the success probability \( P_{\text{succ}} \) and expected distortion \( D \) as defined in Section 2, estimated by

\[
P\text{succ} = \frac{N_{\text{succ}}}{N}, \quad D = \frac{1}{N_{\text{succ}}} \sum_{x_o \in X_{\text{succ}}} D(x_o),
\]

(20)

with the exception that \( D \) here is the conditional average distortion, where conditioning is on success. Indeed, distortion makes no sense for a failure.

If \( D_{\text{max}} = \max_{x_o \in X_{\text{succ}}} D(x_o) \) is the maximum distortion, the operating characteristic function \( P : [0, D_{\text{max}}] \to [0,1] \) measures the probability of success as a function of a given upper bound \( D \) on distortion. For \( D \in [0, D_{\text{max}}] \),

\[
P(D) := \frac{1}{N} |\{ x_o \in X_{\text{succ}} : D(x_o) \leq D \}|.
\]

(21)

This function increases from \( P(0) = 0 \) to \( P(D_{\text{max}}) = P_{\text{succ}} \).

It is difficult to define a fair comparison of distortion targeting attacks to optimality targeting attacks. For the first family, we run a given attack several times over the test set with different target distortion \( \epsilon \). The attack succeeds on image \( x_o \in X \) if it succeeds on any of the runs. For \( x_o \in X_{\text{succ}} \), the distortion \( D(x_o) \) is the minimum distortion over all runs. All statistics are then evaluated as above.

5.2 Datasets, networks, and attacks

5.2.1 MNIST [36]

We consider a simple convolutional network with three convolutional layers and one fully connected layer that we
denote as C4, giving accuracy 0.99. In detail, the first convolutional layer has 64 features, kernel of size 8 and stride 2; the second layer has 128 features, kernel of size 6 and stride 2; the third has also 128 features, but kernel of size 5 and stride 1.

5.2.2 ImageNet
We use the dataset of the NIPS 2017 adversarial competition [37], comprising 1000 images from ImageNet [38]. We use InceptionV3 [39] and ResNetV2-50 [40] networks, with accuracy 0.96 and 0.93 respectively.

5.2.3 Attacks
The following six attacks are benchmarked:

- $L_\infty$ distortion: FGSM [41] and I-FGSM [10].
- $L_2$ distortion: An $L_2$ version of I-FGSM [42], denoted as PGD2 (projected gradient descent).
- Optimality: The $L_2$ version of C&W [8].
- Smooth: Our smooth versions qPGD2 of PGD2 (Section 4.1) and sC&W of C&W (Section 4.2). Note that the smoothness constraint integration differs a lot between qPGD2 and sC&W.

5.2.4 Parameters
On MNIST, we use $\epsilon = 0.3$ for FGSM; $\epsilon = 0.3, \alpha = 0.08$ for I-FGSM; $\epsilon = 5, \alpha = 3$ for PGD2; confidence margin $m = 1$, learning rate $\eta = 0.1$, and initial constant

![Fig. 2](image-url) For a given attack (denoted by an asterisk and bold typeface), the adversarial image with the strongest distortion $D$ over MNIST. In green, the attack succeeds; in red, it fails.
\(c = 15\) (the inverse of \(\lambda\) in (4)) for C\&W. For smoothing, we use Laplacian feature kernel, set \(\alpha = 0.95\), and pre-compute \(L_{\alpha}^{-1}\). On ImageNet, we use \(\epsilon = 0.1255\) for FGSM; \(\epsilon = 0.1255\), \(\alpha = 0.08\) for I-FGSM; \(\epsilon = 5\), \(\alpha = 3\) for PGD2; \(m = 0\), \(\eta = 0.1\), and \(c = 100\) for C\&W. For smoothing, we use Laplacian feature kernel, set \(\alpha = 0.997\), and use 50 iterations of CG. These settings are used in Section 5.5.

5.3 White box scenario

5.3.1 Qualitative results and perceptual evaluation

Figures 2 and 3 show MNIST and ImageNet examples, respectively, focusing on worst cases. Both sC\&W and qPGD2 produce smooth perturbations that look more natural. However, smoothing of qPGD2 is more aggressive especially on MNIST, as these images contain flat black or white areas.

This is due to the “quick and dirty” integration of the smoothness constraint: On some images, the perturbation update \(\hat{s}_{\alpha}(g)\) is weakly correlated with gradient \(g\), which does not help in lowering the classification loss. Consequently, the perturbation becomes stronger in order to succeed. For the same reason, qPGD2 completely fails on natural images like Fig. 3a and c. It consumes way more distortion than PGD2, and although this perturbation is smoother, it becomes visible.

By contrast, the proper integration of the smoothness constraint in sC\&W produces totally invisible perturbation. For images like Fig. 3a or c, sC\&W consumes more distortion than C\&W, but the perturbation remains less visible according to the MAD score. The reason is the “phantom” of the original that is revealed when the perturbation is isolated.

The superior perceptual quality of our smooth adversarial examples is also confirmed quantitatively: On ImageNet, 93% of the images produced by sC\&W have lower MAD score than the ones by C\&W. Figure 4 shows that when the MAD score of sC\&W is greater than the one

---

Fig. 3 Original image \(x_o\) (left), adversarial image \(x_a = x_o + r\) (above) and scaled perturbation \(r\) (below; distortion \(D = \|r\|\) and MAD scores) against InceptionV3 on ImageNet. Scaling maps each perturbation and each color channel independently to [0, 1]. The perturbation \(r\) is indeed smooth like \(x_o\) for sC\&W. a Despite the higher distortion compared to C\&W, the perturbation of sC\&W is totally invisible, even when magnified (cf Fig. 1). b One of the failing examples of [6] that look unnatural to human vision. c One of the examples with the strongest distortion over ImageNet for sC\&W. \(x_o\) is flat along stripes, reducing the dimensionality of the “smooth like \(x_o\)” manifold.
of C&W, it usually happens for very small score values (below 0.1), meaning that both are almost equally imperceptible.

5.3.2 Quantitative results on success and distortion
The global statistics $P_{suc}, D$ are shown in Table 1. Operating characteristics over MNIST and ImageNet are shown in Figs. 5 and 6 respectively.

We observe that our sC&W, with the proper integration via a latent variable (18), improves a lot the original C&W in terms of distortion, while keeping the probability of success roughly the same. This result, consistent in all experiments, is surprising. We would expect a price to be paid for a better invisibility as the smoothing is adding an extra constraint on the perturbation. This price can be rather high in the literature: In order to preserve the success rate, Table 1 of [7] reports an increase of distortion by a factor of 3 when integrating smoothness in the attack. An explanation may be that the smoothing operation of [7] is independent of the input image; while in our case, smoothing is guided by the input.

On the contrary, the “quick and dirty” integration (12) dramatically spoils qPGD2 with big distortion especially on MNIST. This reveals the utmost importance of properly integrating the smoothness constraint. It cannot be just a post-processing filtering of the perturbation.

The price to pay for smoothing is the run-time: using Tensorflow and 50 CG iterations on ImageNet and InceptionV3, sC&W takes 205 s per image on average, while C&W takes 47 s. This is with our own implementation of CG without particular optimization effort. Runtime was not within our objectives.

We further observe that PGD2 outperforms by a vast margin the C&W attack, which is supposed to be close to optimality. This may be due in part to how the Adam optimizer treats $L_2$ norm penalties as studied in [43]. This interesting finding is a result of our new evaluation protocol: C&W internally optimizes its parameter $c = 1/\lambda$ independently per image, while for PGD2, we externally try a small set of target distortions $D$ on the entire dataset. This is visible in Fig. 5, where the operating characteristic is piecewise constant. Our comparison is fair, given that C&W is more expensive.

As already observed in the literature, ResnetV2 is more robust to attacks than InceptionV3: The operating characteristic curves are shifted to the right and increase at a slower rate.

| Table 1 Success probability $P_{suc}$ and average $L_2$ distortion $D$ |
|-----------------------------|-------------------|-------------------|-------------------|-------------------|
|                             | MNIST             | ImageNet          | InceptionV3       | ResNetV2          |
|                             | $P_{suc}$         | $D$               | $P_{suc}$         | $D$               |
| FGSM                        | 0.89              | 5.02              | 0.97              | 5.92              | 0.92              | 8.20              |
| I-FGSM                      | 1.00              | 2.69              | 1.00              | 5.54              | 0.99              | 7.58              |
| PGD2                        | 1.00              | 1.71              | 1.00              | 1.80              | 1.00              | 3.63              |
| C&W                         | 1.00              | 2.49              | 0.99              | 4.91              | 0.99              | 9.84              |
| qPGD2                       | 0.97              | 3.36              | 0.96              | 2.10              | 0.93              | 4.80              |
| sC&W                        | 1.00              | 1.97              | 0.99              | 3.00              | 0.98              | 5.99              |

5.4 Adversarial training
The defender now uses adversarial training [41] to gain robustness against attacks. Yet, the white box scenario still holds: this network is public. The training set comprises images attacked with “step l.l” model [10]. The accuracy of C4 on MNIST (resp. InceptionV3 on ImageNet) is now 0.99 (resp. 0.94).

Table 2 shows interesting results. As expected, FGSM is defeated in all cases, while average distortion of all

\footnote{Model taken from https://github.com/tensorflow/models/tree/master/research/adv_imagenet_models}
attacks is increased in general. What is unexpected is that on MNIST, sC&W remains successful while the probability of C&W drops. On ImageNet however, it is the probability of the smooth versions qPGD$_2$ and sC&W that drops. I-FGSM is also defeated in this case, in the sense that average distortion increases too much.

5.5 Transferability

This section investigates the transferability of the attacks under the following scenario: the attacker has now a partial knowledge about the network. For instance, he/she knows that the defender chose a variant of InceptionV3, but this variant is not public so he/she attacks InceptionV3 instead. Also, this time, he/she is not allowed to test different distortion targets. The results are shown in Table 3.

The first variant uses a bilateral filter (with standard deviations 0.5 and 0.2 in the domain and range kernel respectively; cf Appendix 1) before feeding the network. This does not really prevent the attacks. PGD$_2$ remains a very powerful attack if the distortion is large enough. Smoothing makes the attack less effective, but the perturbations are less visible. The second variant uses the adversarially trained InceptionV3, which is, on the contrary, a very efficient counter-measure under this scenario.

Figure 7 shows the operating characteristics of C&W and sC&W corresponding to the bilateral filter results of Table 3. We see that within a distortion budget of 5, sC&W succeeds with 67% probability, whereas C&W with 50%. Yet, at larger distortion budgets, C&W keeps on forging more adversarial images whereas sC&W stops making progress. This is understandable: C&W creates strong artifacts clearly visible when the distortion is larger or equal to 5, as shown in Fig. 3. The upfront defense filters out some of these strong perturbations, but the rest remain successful. These images are adversarial by definition, yet not useful in practice because they are too much distorted.
### Table 2 Success probability and average $L_2$ distortion $\mathcal{D}$ when attacking networks adversarially trained against FGSM

| Attack | $P_{\text{succ}}$ | $\mathcal{D}$ | $P_{\text{succ}}$ | $\mathcal{D}$ |
|--------|-------------------|----------------|-------------------|----------------|
| FGSM   | 0.15              | 4.53           | 0.06              | 6.40           |
| I-FGSM | 1.00              | 3.48           | 0.97              | 29.94          |
| PGD$_2$| 1.00              | 2.52           | 1.00              | 3.89           |
| C&W    | 0.93              | 3.03           | 0.95              | 6.43           |
| qPGD$_2$| 0.99             | 2.91           | 0.69              | 7.86           |
| sC&W   | 0.99              | 2.39           | 0.75              | 6.22           |

### Table 3 Success probability and average $L_2$ distortion $\mathcal{D}$ of attacks on variants of InceptionV3 under transferability

| Bilateral filter | Adv. training | $P_{\text{succ}}$ | $\mathcal{D}$ | $P_{\text{succ}}$ | $\mathcal{D}$ |
|-----------------|---------------|-------------------|----------------|-------------------|----------------|
| FGSM            |               | 0.77              | 5.13           | 0.04              | 10.20          |
| I-FGSM          |               | 0.82              | 5.12           | 0.02              | 10.10          |
| PGD$_2$         |               | 1.00              | 5.14           | 0.12              | 10.26          |
| C&W             |               | 0.82              | 4.75           | 0.02              | 10.21          |
| qPGD$_2$        |               | 0.95              | 5.13           | 0.01              | 10.17          |
| sC&W            |               | 0.68              | 2.91           | 0.01              | 4.63           |

### 6 Conclusion

Smoothing helps masking the adversarial perturbation by shaping it “like” the input image. However, this rule holds only when smoothness is properly integrated in the attack. Filtering the perturbation by post-processing is not a sound idea, even if it is done in accordance with the original image, even if it is done at each attack iteration. A sounder integration is to inject smoothness as a constraint inside the loss function.

It is impressive how sC&W improves upon C&W in terms of distortion and imperceptibility at the same time while maintaining the same success rate. To our knowledge and as far as a white box scenario is considered, this is the first time smoothness comes for free from this viewpoint. Yet, a price to be paid is the larger complexity.

Smoothing allows the attacker to delude more robust networks thanks to larger distortions while still being invisible. However, its impact on transferability is mitigated. The question raised in the introduction is still open: Fig. 1 shows that a human does not make the difference between the input image and its adversarial example even with magnification. This does not prove that an algorithm will not detect some statistical evidence.

### Appendix 1

#### Adversarial magnification

Given a single-channel image $x : \Omega \rightarrow \mathbb{R}$ as input, its adversarial magnification $\text{mag}(x) : \Omega \rightarrow \mathbb{R}$ is defined as the following local normalization operation

$$\text{mag}(x) := \frac{x - \mu_x(x)}{\beta \sigma_x(x) + (1 - \beta) \sigma_\Omega(x)},$$

where $\mu_x(x)$ and $\sigma_x(x)$ are the local mean and standard deviation of $x$ respectively, and $\sigma_\Omega(x) \in \mathbb{R}^+$ is the global standard deviation of $x$ over $\Omega$. Parameter $\beta \in [0, 1]$ determines how much local variation is magnified in $x$.

In our implementation, $\mu_x(x) = b(x)$, the bilateral filtering of $x$ [44]. It applies a local kernel at each point $p \in \Omega$ that is the product of a domain and a range Gaussian kernel. The domain kernel measures the geometric proximity of every point $q \in \Omega$ to $p$ as a function of $|p - q|$, and the range kernel measures the photometric similarity of every point $q \in \Omega$ to $p$ as a function $|x(p) - x(q)|$. On the other hand, $\sigma_x(x) = b_x((x - \mu_x(x))^2)^{1/2}$. Here, $b_x$ is a guided version of the bilateral filter, where it is the reference image $x$ rather than $(x - \mu_x(x))^2$ that is used in the range kernel.

When $x : \Omega \rightarrow \mathbb{R}^d$ is a $d$-channel image, we apply all the filters independently per channel, but photometric similarity is just one scalar per point as a function of the Euclidean distance $\|x(p) - x(q)\|$ measured over all $d$ channels.

In Fig. 1, $\beta = 0.8$. The standard deviation of both the domain and range Gaussian kernels is 5.

### Acknowledgements

Not applicable.

### Authors’ contributions

The authors declare equal contributions. The authors read and approved the final manuscript.
