Smooth Adversarial Examples

Hanwei Zhang, Yannis Avrithis, Teddy Furon, Laurent Amsaleg

Smooth Adversarial Examples

Hanwei Zhang Yannis Avrithis Teddy Furon Laurent Amsaleg
Univ Rennes, Inria, CNRS, IRISA

Abstract

This paper investigates the visual quality of the adversarial examples. Recent papers propose to smooth the perturbations to get rid of high frequency artefacts. 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.

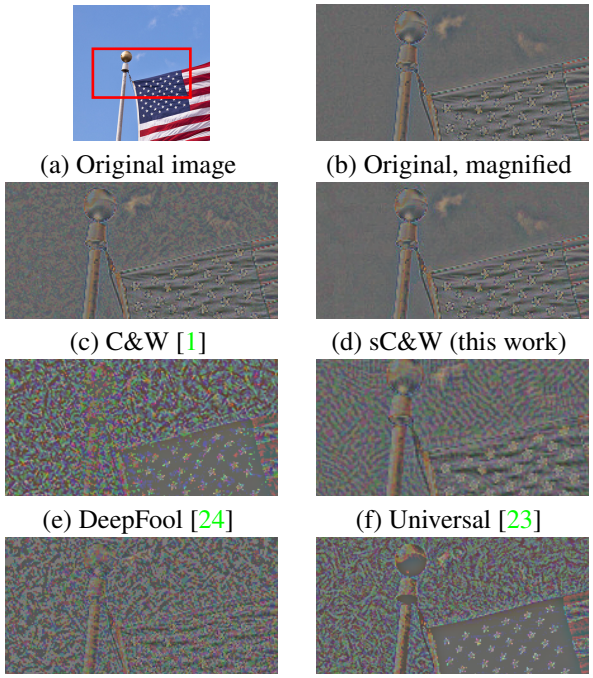
This operation relies on Laplacian smoothing, well-known in graph signal processing, which we integrate in the attack pipeline. We benchmark several attacks with and without smoothing under a white-box scenario and evaluate their transferability. Despite the additional constraint of smoothness, our attack has the same probability of success at lower distortion.

1. Introduction

Adversarial examples where introduced by Szegedy *et al*. [34] 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 [17]. 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 [30].

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 A 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 [23] in our Fig. 1(f), revealing a universal adversarial perturbation.



(a) Original image (b) Original, magnified
(c) C&W [1] (d) sC&W (this work)
(e) DeepFool [24] (f) Universal [23]
(g) FGSM [6] (h) I-FGSM [15]

Figure 1. Given a single input image, our *adversarial magnification* (*cf*. Appendix A) reveals the effect of a potential adversarial perturbation. We show (a) the original image followed by (b) its own magnified version as well as (c)-(h) magnified versions of adversarial examples generated by different attacks. Our *smooth adversarial example* (d) is invisible even when magnified.

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" [37] or being smooth [10, 8], our adversarial examples are photorealistic, low-distortion, and virtually invisible even under magnification. This is evident by comparing our magnified example in Fig. 1(d) to the magnified original in Fig. 1(b).

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 is not only competitive but outperforms Carlini & Wagner [1], from which our own attack differs basically by a smoothness penalty.

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; and
4. 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 section 6. Our *adversarial magnification* used to generate Fig. 1 is specified in Appendix A.

2. Problem formulation and related work

Let us denote by $\mathbf{x} \in \mathcal{X} := [0,1]^{n \times d}$ 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 $\mathbf{x}$ to an output $\mathbf{y} = \mathsf{f}(\mathbf{x}) \in \mathbb{R}^k$ which contains the *logits* of k classes. It is typically followed by $\operatorname{softmax}$ and cross-entropy loss at supervised training or by $\arg\max$ at test time. An input $\mathbf{x}$ with logits $\mathbf{y} = \mathsf{f}(\mathbf{x})$ is correctly classified if the *prediction* $\mathsf{p}(\mathbf{x}) := \arg\max_i y_i$ equals the true label of $\mathbf{x}$.

The attacker mounts a *white-box* attack that is specific to f, public and known. The attack modifies an original image $\mathbf{x}_o \in \mathcal{X}$ with given true label $t \in \{1, \dots, k\}$ into an adversarial example $\mathbf{x}_a \in \mathcal{X}$, which may be incorrectly classified by the network, that is $\mathsf{p}(\mathbf{x}_a) \neq t$, although it looks similar to the original $\mathbf{x}_o$. The latter is often expressed by a small L_2 *distortion* $\|\mathbf{x}_a - \mathbf{x}_o\|$.

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.

Target Distortion. This family gathers attacks targeting a distortion ϵ given as an input parameter. Examples are early attacks like Fast Gradient Sign Method (FGSM) [6] and Iterative-FGSM (I-FGSM) [15]. Their performance is then measured by the *probability of success* $P_{\text{suc}} := \mathbb{P}(\mathsf{p}(\mathbf{x}_a) \neq t)$ as a function of ϵ.

Target Success. This family gathers attacks that always succeed in misclassifying $\mathbf{x}_a$, at the price of a possible large distortion. DeepFool [24] is a typical example. Their performance is then measured by the *expected distortion* $\overline{D} := \mathbb{E}(\|\mathbf{x}_a - \mathbf{x}_o\|)$.

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

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

$$\min_{\mathbf{x} \in \mathcal{X} : \mathsf{p}(\mathbf{x}) \neq t} \|\mathbf{x} - \mathbf{x}_o\|. \tag{1}$$

Szegedy *et al*. [34] approximate this constrained minimization problem by a Lagrangian formulation

$$\min_{\mathbf{x} \in \mathcal{X}} \lambda \|\mathbf{x} - \mathbf{x}_o\|^2 + \ell(\mathsf{f}(\mathbf{x}), t). \tag{2}$$

Parameter λ controls the trade-off between the distortion and the classification loss. Szegedy *et al*. [34] carry out this optimization by box-constrained L-BFGS.

The attack of Carlini & Wagner [1], denoted C&W in the sequel, pertains to this approach. A change of variable eliminates the box constraint: $\mathbf{x} \in \mathcal{X}$ is replaced by $\sigma(\mathbf{w})$, where $\mathbf{w} \in \mathbb{R}^{n \times d}$ and σ is the element-wise sigmoid function. A margin is introduced: an *untargeted attack* makes the logit y_t 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 [4] (where $m = 1$), the loss function ℓ is then defined as

$$\ell(\mathbf{y}, t) := [y_t - \max_{i \neq t} y_i + m]_+, \tag{3}$$

where $[\cdot]_+$ denotes the positive part. The C&W attack uses the Adam optimizer [14] to minimize the functional

$$J(\mathbf{w}, \lambda) := \lambda \|\sigma(\mathbf{w}) - \mathbf{x}_o\|^2 + \ell(\mathsf{f}(\sigma(\mathbf{w})), t). \tag{4}$$

When the margin is reached, loss $\ell(\mathbf{y}, t)$ (3) vanishes and the distortion term pulls $\sigma(\mathbf{w})$ back towards $\mathbf{x}_o$, 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 λ 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*. [40] 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*. [10] propose a harmonic adversarial attack where perturbations are very smooth gradient-like images. Guo *et al*. [8] 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 $\mathbf{x}_o$ but smooth wherever $\mathbf{x}_o$ is, *e.g*. on background regions. It is not just smooth but photorealistic, because its smoothness pattern is guided by the input image.

An analogy becomes evident with *digital watermarking* [28]. 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 [3]. Its textured areas and its contours can hide a lot of watermarking power, but the flat areas can not be modified without producing noticeable artefacts. 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*. [22] 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 [27, 20] are 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* [32]. 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* [38, 13], a classical operator in *graph signal processing* [29, 31], which we adapt to images here. Section 4 uses it generate a smooth perturbation guided by the original input image.

Graph. Laplacian smoothing builds on a weighted undirected graph whose n vertices correspond to the n pixels of the input image $\mathbf{x}_o$. The i-th vertex of the graph is associated with feature $\mathbf{x}_i \in [0,1]^d$ that is the i-th row of $\mathbf{x}_o$, that is, $\mathbf{x}_o = [\mathbf{x}_1, \dots, \mathbf{x}_n]^\top$. Matrix $\mathbf{p} \in \mathbb{R}^{n\times 2}$ denotes the spatial coordinates of the n pixels in the image, and similarly $\mathbf{p} = [\mathbf{p}_1, \dots, \mathbf{p}_n]^\top$. An edge (i,j) of the graph is associated with weight $w_{ij} \geq 0$, giving rise to an $n \times n$ symmetric adjacency matrix $\mathbf{W}$, for instance defined as

$$w_{ij} := \begin{cases} \mathsf{k_f}(\mathbf{x}_i, \mathbf{x}_j)\mathsf{k_s}(\mathbf{p}_i, \mathbf{p}_j), & \text{if } i \neq j \\ 0, & \text{if } i = j \end{cases} \tag{5}$$

for $i, j \in \{1, \dots, n\}$, where $\mathsf{k_f}$ is a *feature kernel* and $\mathsf{k_s}$ 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 $\mathbf{W}$. We further define the $n \times n$ *degree matrix* $\mathbf{D} := \mathrm{diag}(\mathbf{W}\mathbf{1}_n)$ where $\mathbf{1}_n$ is the all-ones n-vector.

Regularization [38]. Now, given a new signal $\mathbf{z} \in \mathbb{R}^{n\times d}$ on this graph, the objective of graph smoothing is to find the output signal $\mathsf{s}_\alpha(\mathbf{z}) := \arg\min_{\mathbf{r}\in\mathbb{R}^{n\times d}} \phi_\alpha(\mathbf{r}, \mathbf{z})$ with

$$\phi_\alpha(\mathbf{r}, \mathbf{z}) := \frac{\alpha}{2} \sum_{i,j} w_{ij} \left\| \hat{\mathbf{r}}_i - \hat{\mathbf{r}}_j \right\|^2 + (1-\alpha) \left\| \mathbf{r} - \mathbf{z} \right\|_F^2 \tag{6}$$

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

Filtering. If we symmetrically normalize matrix $\mathbf{W}$ as $\mathcal{W} := \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}$ and define the $n \times n$ *regularized Laplacian* matrix $\mathcal{L}_\alpha := (\mathbf{I}_n - \alpha\mathcal{W})/(1-\alpha)$, then the expression (6) simplifies to the following quadratic form:

$$\phi_\alpha(\mathbf{r}, \mathbf{z}) = (1-\alpha)\,\mathrm{tr}\left(\mathbf{r}^\top \mathcal{L}_\alpha \mathbf{r} - 2\mathbf{z}^\top \mathbf{r} + \mathbf{z}^\top \mathbf{z}\right). \tag{7}$$

This reveals, by letting the derivative $\partial\phi/\partial\mathbf{r}$ vanish independently per column, that the smoothed signal is simply:

$$\mathsf{s}_\alpha(\mathbf{z}) = \mathcal{L}_\alpha^{-1}\mathbf{z}. \tag{8}$$

This is possible because matrix $\mathcal{L}_\alpha$ is positive-definite. Parameter α controls the bandwidth of the smoothing: function s_α is the all-pass filter for $\alpha = 0$ and becomes a strict 'low-pass' filter when $\alpha \to 1$ [11].

Variants of the model above have been used for instance for interactive image segmentation [7, 13, 36], transductive semi-supervised classification [41, 38], and ranking on

manifolds [39, 12]. Input $\mathbf{z}$ 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.

Normalization. Contrary to applications like interactive segmentation or semi-supervised classification [38, 13], $\mathbf{z}$ 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 $\mathcal{L}_\alpha^{-1}$ is seen as a spatial filter, we therefore row-wise normalize it to one in order to preserve the dynamic range of $\mathbf{z}$:

$$\hat{\mathsf{s}}_\alpha(\mathbf{z}) := \operatorname{diag}(\mathsf{s}_\alpha(\mathbf{1}_n))^{-1}\mathsf{s}_\alpha(\mathbf{z}). \tag{9}$$

The *normalized smoothing function* $\hat{\mathsf{s}}_\alpha$ of course depends on $\mathbf{x}_o$. We omit this from notation but we say $\hat{\mathsf{s}}_\alpha$ is *smoothing guided* by $\mathbf{x}_o$ and the output is *smooth like* $\mathbf{x}_o$.

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 $\mathbf{x}_o$, which is achieved by smoothing operations guided by $\mathbf{x}_o$. This section integrates smoothness into attacks targeting distortion (section 4.1) and attacks targeting optimality (section 2.1).

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

$$\mathbf{g} = \nabla_{\mathbf{x}}\ell(\mathsf{f}(\mathbf{x}_a^{(k)}), t), \tag{10}$$

$$\mathbf{x}_a^{(k+1)} = \mathsf{c}\left(\mathbf{x}_a^{(k)} - \mathsf{n}(\mathbf{g})\right), \tag{11}$$

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

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

$$\mathbf{x}_a^{(k+1)} = \mathsf{c}\left(\mathbf{x}_a^{(k)} - \mathsf{n}(\hat{\mathsf{s}}_\alpha(\mathbf{g}))\right). \tag{12}$$

This approach can be seen as a projected gradient descent on the manifold of perturbations that are smooth like $\mathbf{x}_o$.

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) [1]. However, instead of representing the perturbation signal $\mathbf{r} := \mathbf{x} - \mathbf{x}_o$ implicitly as a function $\sigma(\mathbf{w}) - \mathbf{x}_o$ of another parameter $\mathbf{w}$, we express the objective explicitly as a function of variable $\mathbf{r}$, as in the original formulation of (2) in [34]. We make this choice because we need to directly process the perturbation $\mathbf{r}$ independently of $\mathbf{x}_o$. On the other hand, we now need the element-wise clipping function $\mathsf{c}(\mathbf{x}) := \min([\mathbf{x}]_+, 1)$ to satisfy the constraint $\mathbf{x} = \mathbf{x}_o + \mathbf{r} \in \mathcal{X}$ (2). Our problem is then

$$\min_{\mathbf{r}} \quad \lambda\left\|\mathbf{r}\right\|^2 + \ell(\mathsf{f}(\mathsf{c}(\mathbf{x}_o + \mathbf{r})), t), \tag{13}$$

where $\mathbf{r}$ is unconstrained in $\mathbb{R}^{n\times d}$.

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 $\mathbf{x}_o$ into account and impose a similar structure on $\mathbf{r}$. Representing the pairwise relations by a graph as discussed in section 3, a straightforward choice is to introduce a pairwise loss term

$$\mu\sum_{i,j} w_{ij}\left\|\hat{\mathbf{r}}_i - \hat{\mathbf{r}}_j\right\|^2 \tag{14}$$

into (13), where we recall that w_{ij} are the elements of the adjacency matrix $\mathbf{W}$ of $\mathbf{x}_o$, $\hat{\mathbf{r}} := \mathbf{D}^{-1/2}\mathbf{r}$ and $\mathbf{D} := \operatorname{diag}(\mathbf{W}\mathbf{1}_n)$. A problem is that the spatial kernel is typically narrow to capture smoothness only locally. Even if parameter μ is large, it would take a lot of iterations for the information to propagate globally, each iteration needing a forward and backward pass through the network.

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

$$\min_{\mathbf{r},\mathbf{z}} \quad \mu\phi_\alpha(\mathbf{r},\mathbf{z}) + \lambda\left\|\mathbf{r}\right\|^2 + \ell(\mathsf{f}(\mathsf{c}(\mathbf{x}_o + \mathbf{r})), t), \tag{15}$$

where ϕ is defined by (6). In words, $\mathbf{z}$ represents an unconstrained perturbation, while $\mathbf{r}$ should be close to $\mathbf{z}$, smooth like $\mathbf{x}_o$, small, and such that the perturbed input $\mathbf{x}_o + \mathbf{r}$ 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:

$$\min_{\mathbf{r},\mathbf{z}} \quad \lambda\left\|\mathbf{r}\right\|^2 + \ell(\mathsf{f}(\mathsf{c}(\mathbf{x}_o + \mathbf{r})), t) \tag{16}$$

$$\text{subject to} \quad \mathbf{r} = \hat{\mathsf{s}}_\alpha(\mathbf{z}), \tag{17}$$

where $\hat{\mathsf{s}}_\alpha$ is defined by (9). During optimization, every iterate of this perturbation $\mathbf{r}$ is smooth like $\mathbf{x}_o$.

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

$$\min_{\mathbf{z}} \quad \lambda \left\|\hat{\mathsf{s}}_\alpha(\mathbf{z})\right\|^2 + \ell(\mathsf{f}(\mathsf{c}(\mathbf{x}_o + \hat{\mathsf{s}}_\alpha(\mathbf{z}))), t). \tag{18}$$

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

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

$$\nabla_{\mathbf{z}} L(\mathbf{z}) = \mathbf{J}_{\hat{\mathsf{s}}_\alpha}(\mathbf{z})^\top \cdot \nabla_{\mathbf{x}} \ell(\mathsf{f}(\mathsf{c}(\mathbf{x}_o + \hat{\mathsf{s}}_\alpha(\mathbf{z}))), t), \tag{19}$$

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

Matrix $\mathcal{L}_\alpha$ is fixed during optimization, depending only on input $\mathbf{x}_o$. For small images like in the MNIST dataset [18], it can be inverted: function $\hat{\mathsf{s}}_\alpha$ is really a matrix multiplication. For larger images, we use the *conjugate gradient* (CG) method [25] to solve the set of linear systems $\mathcal{L}_\alpha\mathbf{r} = \mathbf{z}$ for $\mathbf{r}$ given $\mathbf{z}$. Again, this is possible because matrix $\mathcal{L}_\alpha$ is positive-definite, and indeed it is the most common solution in similar problems [7, 2, 12]. At each iteration, one computes a product of the form $\mathbf{v} \mapsto \mathcal{L}_\alpha\mathbf{v}$, which is efficient because $\mathcal{L}_\alpha$ is sparse. In the backward pass, one can either use CG on the gradient, or auto-differentiate through the forward CG iterations. These options have the same complexity. We choose the latter.

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 [1] therefore argue that the numerical solver of problem (13) suffers from getting stuck in flat spots: when a pixel of the perturbed input $\mathbf{x}_o + \mathbf{r}$ 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 L_2 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 *untargetted*, as defined by loss function (3).

Table 1. Success probability P_{suc} and average L_2 distortion $\overline{D}$.

	MNIST		ImageNet			
	C4		InceptionV3		ResNetV2	
	P_{suc}	$\overline{D}$	P_{suc}	$\overline{D}$	P_{suc}	$\overline{D}$
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
PGD_2	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
sPGD_2	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

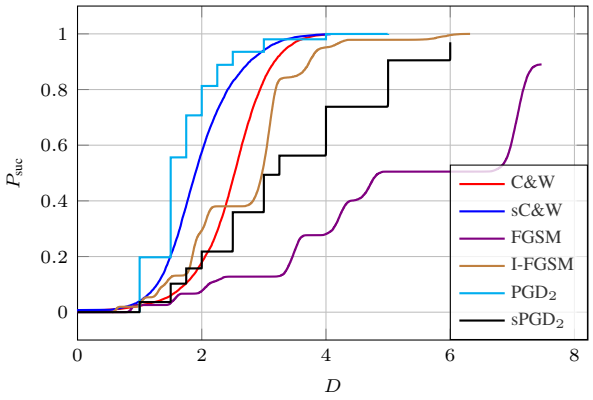


Figure 2. Operating characteristics of the attacks over MNIST. Attacks PGD_2 and sPGD_2 are tested with target distortion $D \in [1, 6]$.

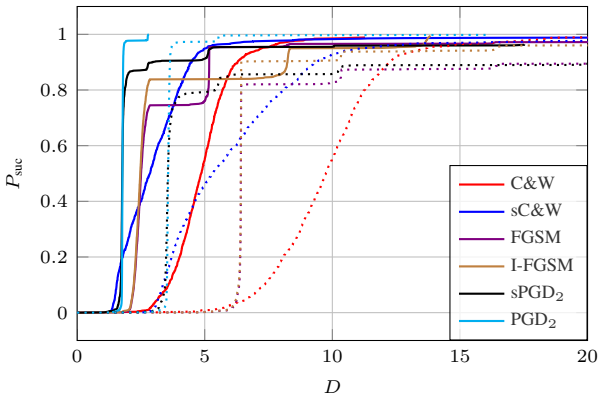


Figure 3. Operating characteristics over ImageNet attacking InceptionV3 (solid lines) and ResNetV2-50 (dotted lines).

5.1. Evaluation protocol

We evaluate the strength of an attack by two global statistics and by 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_{suc} be the subset

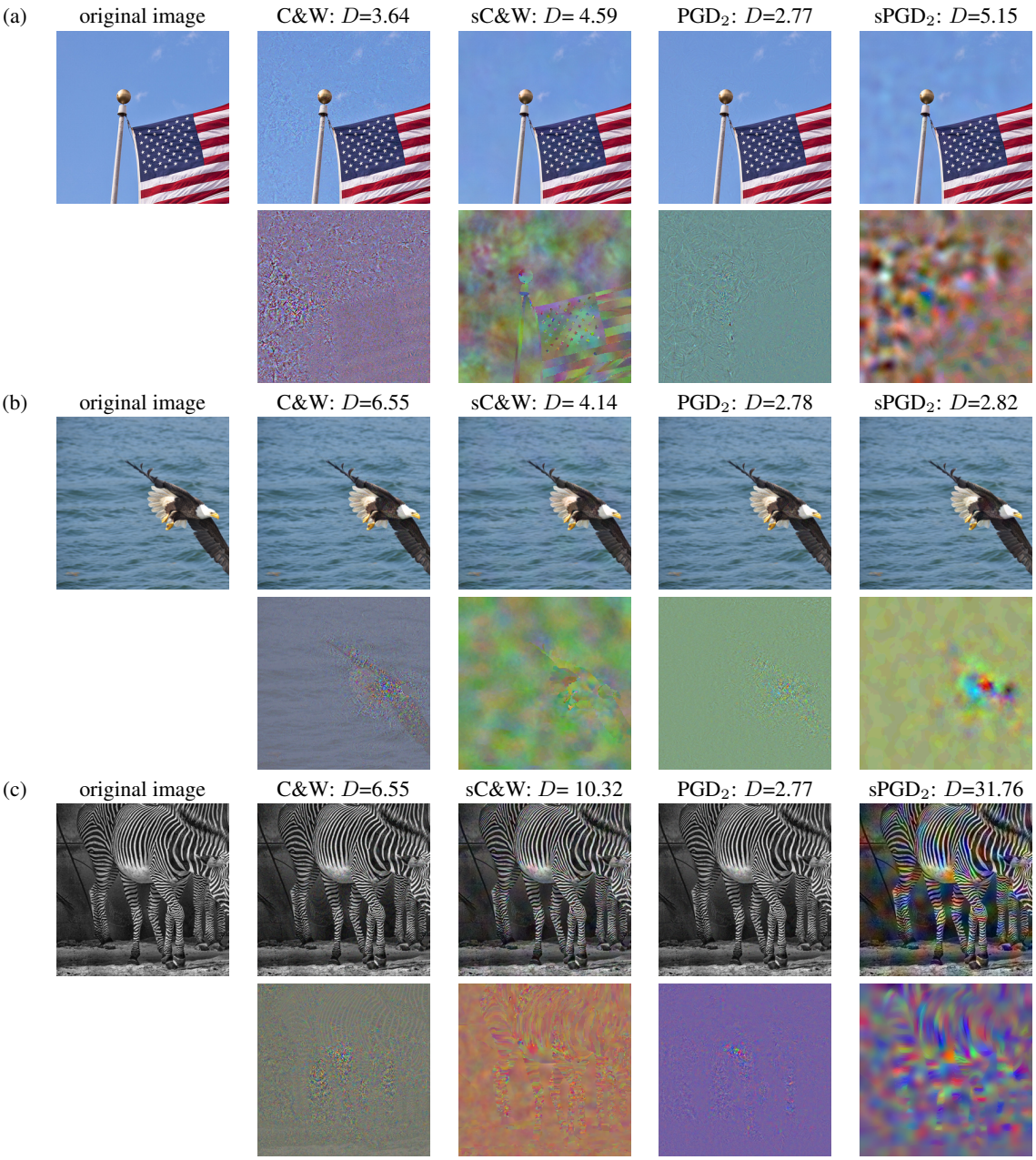


Figure 4. Original image $\mathbf{x}_o$ (left), adversarial image $\mathbf{x}_a = \mathbf{x}_o + \mathbf{r}$ (above) and scaled perturbation $\mathbf{r}$ (below; distortion $D = \|\mathbf{r}\|$) against InceptionV3 on ImageNet. Scaling maps each perturbation and each color channel independently to $[0, 1]$. The perturbation $\mathbf{r}$ is indeed smooth like $\mathbf{x}_o$ 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 [10] that look unnatural to human vision. (c) One of the examples with the strongest distortion over ImageNet for sC&W: $\mathbf{x}_o$ is flat along stripes, reducing the dimensionality of the 'smooth like $\mathbf{x}_o$' manifold.

of X with $N_{\text{suc}} := |X_{\text{suc}}|$ where the attack succeeds and let $D(\mathbf{x}_o) := \|\mathbf{x}_a - \mathbf{x}_o\|$ be the distortion for image $\mathbf{x}_o \in X_{\text{suc}}$.

The global statistics are the *success probability* P_{suc} and *expected distortion* $\overline{D}$ as defined in section 2, estimated by

$$P_{\text{suc}} = \frac{N_{\text{suc}}}{N}, \quad \overline{D} = \frac{1}{N_{\text{suc}}} \sum_{\mathbf{x}_o \in X_{\text{suc}}} D(\mathbf{x}_o), \qquad (20)$$

with the exception that $\overline{D}$ here is the *conditional average distortion*, where conditioning is on success. Indeed, distortion makes no sense for a failure.

If $D_{\max} = \max_{\mathbf{x}_o \in X_{\text{suc}}} D(\mathbf{x}_o)$ is the maximum distortion, the *operating characteristic* function $\mathsf{P} : [0, D_{\max}] \to [0, 1]$ measures the probability of success as a function of a

given upper bound D on distortion. For $D \in [0, D_{\max}]$,

$$\mathsf{P}(D) := \frac{1}{N} |\{\mathbf{x}_o \in X_{\text{suc}} : D(\mathbf{x}_o) \leq D\}|. \quad (21)$$

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

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 ϵ. The attack succeeds on image $\mathbf{x}_o \in X$ if it succeeds on any of the runs. For $\mathbf{x}_o \in X_{\text{suc}}$, the distortion $D(\mathbf{x}_o)$ is the minimum distortion over all runs. All statistics are then evaluated as above.

5.2. Datasets, networks, and attacks

MNIST [19]. 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.

ImageNet. We use the dataset of the NIPS 2017 adversarial competition [16], comprising 1,000 images from ImageNet [5]. We use InceptionV3 [33] and ResNetV2-50 [9] networks, with accuracy 0.96 and 0.93 respectively.

Attacks. The following six attacks are benchmarked:

- L_∞ distortion: FGSM [6] and I-FGSM [15].
- L_2 distortion: an L_2 version of I-FGSM [26], denoted as PGD_2 (projected gradient descent).
- Optimality: The L_2 version of C&W [1].
- Smooth: our smooth versions sPGD_2 of PGD_2 (sect. 4.1) and sC&W of C&W (sect. 4.2).

Parameters. On MNIST, we use $\epsilon = 0.3$ for FGSM; $\epsilon = 0.3, \alpha = 0.08$ for I-FGSM; $\epsilon = 5, \alpha = 3$ for PGD_2; confidence margin $m = 1$, learning rate $\eta = 0.1$, and initial constant $c = 15$ (the inverse of λ in (4)) for C&W. For smoothing, we use Laplacian feature kernel, set $\alpha = 0.95$, and pre-compute $\mathcal{L}_\alpha^{-1}$. On ImageNet, we use $\epsilon = 0.1255$ for FGSM; $\epsilon = 0.1255, \alpha = 0.08$ for I-FGSM; $\epsilon = 5, \alpha = 3$ for PGD_2; $m = 0$, $\eta = 0.1$, 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 sect. 5.4.

5.3. White box scenario

The global statistics $P_{\text{suc}}, \overline{D}$ are shown in Table 1. Operating characteristics over MNIST and ImageNet are shown in Figures 2 and 3 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 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. All results clearly show the opposite. On the contrary, the 'quick and dirty' integration (12) dramatically spoils sPGD_2 with big distortion especially on MNIST. This reveals that attacks behave in different ways under the new constraint.

We further observe that PGD_2 outperforms by a vast margin C&W, 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 [21]. C&W internally optimizes its parameter $c = 1/\lambda$ independently per image, while for PGD_2 we externally try a small set of target distortions D on the entire dataset. This is visible in Fig. 2, where the operating characteristic is piecewise constant. This interesting finding is a result of our new evaluation protocol. Our comparison is fair, given that C&W is more expensive.

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

To further understand the different behavior of C&W and PGD_2 under smoothness, Figures 5 and 4 show MNIST and ImageNet examples respectively, focusing on worst cases. Both sPGD_2 and sC&W produce smooth perturbations that look more natural. However, smoothing of sPGD_2 is more aggressive especially on MNIST, as these images contain flat black or white areas. The perturbation update $\hat{\mathsf{s}}_\alpha(\mathbf{g})$ is then weakly correlated with gradient $\mathbf{g}$, which is not efficient to lower the classification loss. In natural images, except for *worst cases* like Fig 4(c), the perturbation of sC&W is totally invisible. The reason is the 'phantom' of the original that is revealed when the perturbation is isolated.

Adversarial training. The defender now uses adversarial training [6] 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 [15][1]. The accuracy of C4 over MNIST (resp. InceptionV3 over 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 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 on the other hand, it is the probability of the smooth versions sPGD_2 and sC&W that drops. I-FGSM is also defeated in this case, in the sense that average distortion increases too much.

5.4. 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

[1]Model taken from: https://github.com/tensorflow/models/tree/master/research/adv_imagenet_models

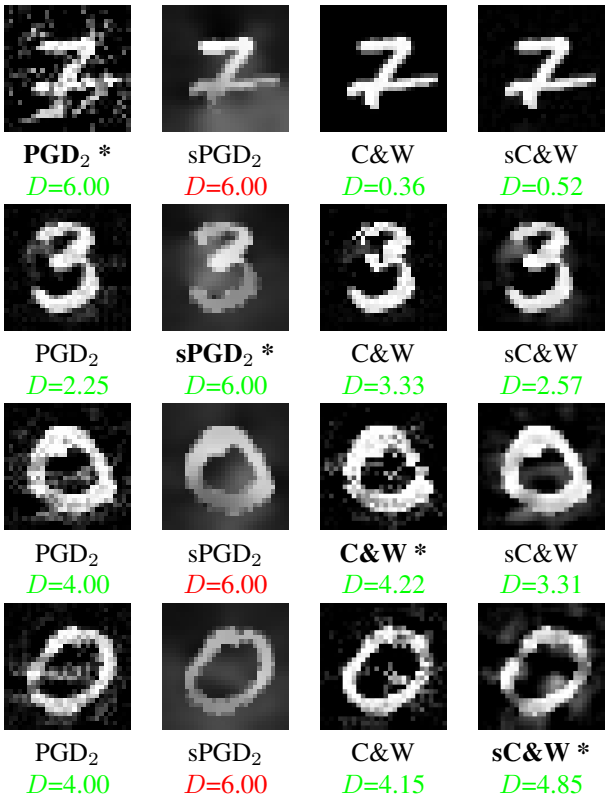


Figure 5. For a given attack (denoted by * and bold typeface), the adversarial image with the strongest distortion D over MNIST. In green, the attack succeeds; in red, it fails.

Table 2. Success probability and average L_2 distortion $\overline{D}$ when attacking networks adversarially trained against FGSM.

	MNIST - C4		ImageNet - InceptionV3	
	P_{suc}	$\overline{D}$	P_{suc}	$\overline{D}$
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
sPGD$_2$	0.99	2.94	0.69	7.86
sC&W	0.99	2.39	0.75	6.22

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

	Bilateral filter		Adv. training	
	P_{suc}	$\overline{D}$	P_{suc}	$\overline{D}$
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
sPGD$_2$	0.95	5.13	0.01	10.17
sC&W	0.68	2.91	0.01	4.63

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 deviation 0.5 and 0.2 in the domain and range kernel respectively; *cf*. appendix A) 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 does not improve the statistics 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.

6. Conclusion

Smoothing helps mask the adversarial perturbation, when it is 'like' the input image. It allows the attacker to delude more robust networks thanks to larger distortions while still being invisible. However, its impact on transferability is mitigated. While clearly not every attack is improved by such smoothing, which was not our objective, it is impressive how sC&W improves upon C&W in terms of distortion and imperceptibility at the same time.

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.

A. Adversarial magnification

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

$$\mathsf{mag}(\mathbf{x}) := \frac{\mathbf{x} - \mu_{\mathbf{x}}(\mathbf{x})}{\beta\sigma_{\mathbf{x}}(\mathbf{x}) + (1-\beta)\sigma_{\Omega}(\mathbf{x})}, \tag{22}$$

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

In our implementation, $\mu_{\mathbf{x}}(\mathbf{x}) = \mathsf{b}(\mathbf{x})$, the *bilateral filtering* of $\mathbf{x}$ [35]. 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 $|\mathbf{x}(p) - \mathbf{x}(q)|$. On the other hand, $\sigma_{\mathbf{x}}(\mathbf{x}) = \mathsf{b}_{\mathbf{x}}((\mathbf{x} - \mu_{\mathbf{x}}(\mathbf{x}))^2)^{-1/2}$, where $\mathsf{b}_{\mathbf{x}}$ is a *guided* version of the bilateral filter, where it is the reference image $\mathbf{x}$ rather than $(\mathbf{x} - \mu_{\mathbf{x}}(\mathbf{x}))^2$ that is used in the range kernel.

When $\mathbf{x} : \Omega \to \mathbb{R}^d$ 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 $\|\mathbf{x}(p) - \mathbf{x}(q)\|$ measured over all d channels.

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

References

[1] N. Carlini and D. Wagner. Towards evaluating the robustness of neural networks. In *IEEE Symp. on Security and Privacy*, 2017. 1, 2, 4, 5, 7

[2] S. Chandra and I. Kokkinos. Fast, exact and multi-scale inference for semantic image segmentation with deep Gaussian CRFs. In *ECCV*, 2016. 5

[3] I. Cox, M. Miller, J. Bloom, J. Fridrich, and T. Kalker. *Digital Watermarking*. Morgan Kaufmann Publisher, second edition, 2008. 3

[4] K. Crammer and Y. Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research*, 2(Dec), 2001. 2

[5] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, pages 248–255. Ieee, 2009. 7

[6] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. *arXiv:1412.6572*, 2014. 1, 2, 7

[7] L. Grady. Random walks for image segmentation. *IEEE Trans. PAMI*, 28(11), 2006. 3, 5

[8] C. Guo, J. S. Frank, and K. Q. Weinberger. Low frequency adversarial perturbation. *arXiv:1809.08758*, 2018. 2, 3

[9] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In *ECCV*, pages 630–645. Springer, 2016. 7

[10] W. Heng, S. Zhou, and T. Jiang. Harmonic adversarial attack method. *arXiv:1807.10590*, 2018. 2, 3, 6

[11] A. Iscen, Y. Avrithis, G. Tolias, T. Furon, and O. Chum. Fast spectral ranking for similarity search. In *CVPR*, 2018. 3

[12] A. Iscen, G. Tolias, Y. Avrithis, T. Furon, and O. Chum. Efficient diffusion on region manifolds: Recovering small objects with compact cnn representations. In *CVPR*, 2017. 4, 5

[13] T. H. Kim, K. M. Lee, and S. U. Lee. Generative image segmentation using random walks with restart. In *ECCV*, 2008. 3, 4

[14] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv:1412.6980*, 2015. 2

[15] A. Kurakin, I. Goodfellow, and S. Bengio. Adversarial examples in the physical world. *arXiv:1607.02533*, 2016. 1, 2, 7

[16] A. Kurakin, I. Goodfellow, S. Bengio, Y. Dong, F. Liao, M. Liang, T. Pang, J. Zhu, X. Hu, C. Xie, et al. Adversarial attacks and defences competition. *arXiv:1804.00097*, 2018. 7

[17] A. Kurakin, I. Goodfellow, S. Bengio, Y. Dong, F. Liao, M. Liang, T. Pang, J. Zhu, X. Hu, C. Xie, J. Wang, Z. Zhang, Z. Ren, A. Yuille, S. Huang, Y. Zhao, Y. Zhao, Z. Han, J. Long, Y. Berdibekov, T. Akiba, S. Tokui, and M. Abe. Adversarial attacks and defences competition. *arXiv:1804.00097*, 2018. 1

[18] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proc. of the IEEE*, 86(11), 1998. 5

[19] Y. LeCun, C. Cortes, and C. Burges. Mnist handwritten digit database. *AT&T Labs [Online]. Available: http://yann. lecun. com/exdb/mnist*, 2, 2010. 7

[20] Y. Li, M.-Y. Liu, X. Li, M.-H. Yang, and J. Kautz. A closed-form solution to photorealistic image stylization. *arXiv:1802.06474*, 2018. 3

[21] I. Loshchilov and F. Hutter. Fixing weight decay regularization in adam. *CoRR*, abs/1711.05101, 2017. 7

[22] F. Luan, S. Paris, E. Shechtman, and K. Bala. Deep photo style transfer. *CVPR*, 2017. 3

[23] S. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard. Universal adversarial perturbations. In *CVPR*, pages 86–94, 2017. 1

[24] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *CVPR*, 2016. 1, 2

[25] J. Nocedal and S. Wright. *Numerical optimization*. Springer, 2006. 5

[26] N. Papernot, F. Faghri, N. Carlini, I. Goodfellow, R. Feinman, A. Kurakin, C. Xie, Y. Sharma, T. Brown, A. Roy, A. Matyasko, V. Behzadan, K. Hambardzumyan, Z. Zhang, Y.-L. Juang, Z. Li, R. Sheatsley, A. Garg, J. Uesato, W. Gierke, Y. Dong, D. Berthelot, P. Hendricks, J. Rauber, and R. Long. Technical report on the cleverhans v2.1.0 adversarial examples library. *arXiv:1610.00768*, 2018. 7

[27] G. Puy and P. Pérez. A flexible convolutional solver with application to photorealistic style transfer. *arXiv:1806.05285*, 2018. 3

[28] E. Quiring, D. Arp, and K. Rieck. Forgotten siblings: Unifying attacks on machine learning and digital watermarking. In *2018 IEEE European Symposium on Security and Privacy (EuroS P)*, pages 488–502, April 2018. 3

[29] A. Sandryhaila and J. M. Moura. Discrete signal processing on graphs. IEEE *Trans. on Signal Processing*, 61(7), 2013. 3

[30] M. Sharif, L. Bauer, and M. K. Reiter. On the suitability of l_p-norms for creating and preventing adversarial examples. *arXiv:1802.09653*, 2018. 1

[31] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE Signal Processing Magazine*, 30(3), 2013. 3

[32] J. Su, D. V. Vargas, and S. Kouichi. One pixel attack for fooling deep neural networks. *arXiv:1710.08864*, 2017. 3

[33] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016. 7

[34] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. *arXiv:1312.6199*, 2013. 1, 2, 4

[35] C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. In *ICCV*, 1998. 8

[36] P. Vernaza and M. Chandraker. Learning random-walk label propagation for weakly-supervised semantic segmentation. In *CVPR*, 2017. 3

[37] Z. Zhao, D. Dua, and S. Singh. Generating natural adversarial examples. In *ICLR*, 2018. 2

[38] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf. Learning with local and global consistency. In *NIPS*, 2003. 3, 4

[39] D. Zhou, J. Weston, A. Gretton, O. Bousquet, and B. Schölkopf. Ranking on data manifolds. In *NIPS*, 2003. 4

[40] W. Zhou, X. Hou, Y. Chen, M. Tang, X. Huang, X. Gan, and Y. Yang. Transferable adversarial perturbations. In *ECCV*, 2018. 3

[41] X. Zhu, Z. Ghahramani, and J. Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *ICLR*, 2003. 4