Walking on the Edge: Fast, Low-Distortion Adversarial Examples

Hanwei Zhang[1,2], Yannis Avrithis[1], Teddy Furon[1], and Laurent Amsaleg[1]

[1]Inria, CNRS, IRISA, Univ Rennes
[2]East China Normal University

***Abstract*—Adversarial examples of deep neural networks are receiving ever increasing attention because they help in understanding and reducing the sensitivity to their input. This is natural given the increasing applications of deep neural networks in our everyday lives. When white-box attacks are almost always successful, it is typically only the distortion of the perturbations that matters in their evaluation.**

In this work, we argue that speed is important as well, especially when considering that fast attacks are required by adversarial training. Given more time, iterative methods can always find better solutions. We investigate this *speed-distortion trade-off* in some depth and introduce a new attack called *boundary projection* (BP) that improves upon existing methods by a large margin. Our key idea is that the classification boundary is a manifold in the image space: we therefore quickly reach the boundary and then optimize distortion on this manifold.

***Index Terms*—Adversarial attack, Deep learning**

I. Introduction

ADVERSARIAL EXAMPLES [37] are small, usually imperceptible perturbations of images or other data [9] that can arbitrarily modify a classifier's prediction. They have been extended to other tasks like object detection or semantic segmentation [45], and image retrieval [25], [39]. They are typically generated in a *white-box* setting, where the attacker has full access to the classifier model and uses gradient signals through the model to optimize for the perturbation. They are becoming increasingly important because they reveal the *sensitivity* of neural networks to their input [35], [15], [2] including trivial cases [3], [14] and they easily *transfer* between different models [28], [41].

Adversarial examples are typically evaluated by *probability of success* and *distortion*. In many cases, white-box attacks have probability of success near one, then only distortion matters, as a (weak) measure of imperceptibility and also of the difficulty with which adversarial samples can be detected. The *speed* of an attack is less frequently discussed. The fast single-step FGSM attack [17] produces high-distortion examples where adversarial patterns can easily be recognized. At the other extreme, the *Carlini & Wagner* (C&W) attack [8], considered state of the art, is notoriously expensive. *Decoupling direction and norm* (DDN) [32] has recently shown impressive progress in distortion and mostly in speed.

Speed becomes more important when considering *adversarial training* [17]. This defense, where adversarial examples are used for training, was in fact introduced in the same work as FGSM. This attack remains the most common choice for generating those examples because of its speed. However, adversarial training is easily broken [40] unless a more powerful attack like PGD is used [27].

In this work, we investigate in more depth the *speed-distortion trade-off* in the regime of probability of success near one. We observe that iterative attacks often oscillate across the classification boundary, taking long time to stabilize. We introduce a new attack that rather walks *along* the boundary, as discussed in Sect. I-B. As a result, we improve the state of the art in distortion while keeping the number of iterations at a minimum. Our key idea is that, once we reach the adversarial region near the boundary, the problem becomes *optimization on a manifold* [1], in particular, minimization of the ℓ_2 distortion on a *level set* of the classification loss. When in the adversarial region, we project the distortion gradient on the *tangent space* of this manifold. We do this simply by targeting a particular reduction of the distortion while moving orthogonally to the gradient of the classification loss.

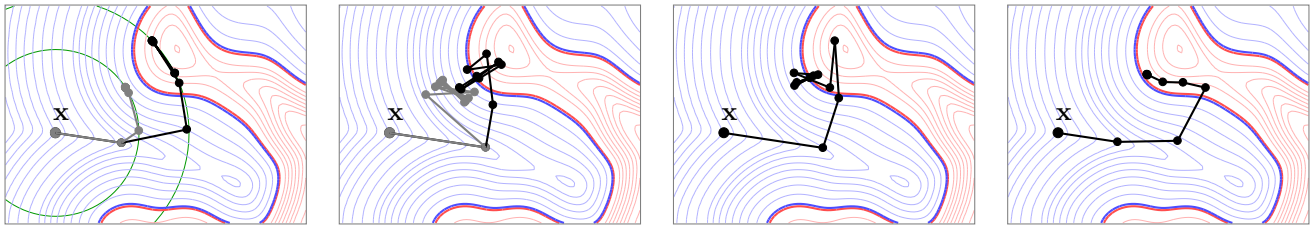
Quantization is another major issue in this literature. Most papers implicitly assume that the output of a white-box attack is a matrix where pixel values are real numbers in $[0, 1]$. Paper [32] is one of the rare works where the output is quantized. We agree with this definition of the problem. Indeed, an adversarial image is above all an image. The goal of an attacker is to publish images deluding the classifier (for instance on the web), and publishing implies compliance with pixels encoded in bytes.

Although adversarial training is not the focus of this work, we do experiment with it to validate that i) our attack is fast enough for this task, and ii) the network gains a better defense when being prepared for a worse attack. Recent improvements in adversarial training [42] are orthogonal to our work, by replacing the common choice of PGD with our attack.

A. Contributions.

To our knowledge, we are the first to

- Study optimization on the *manifold* of the classification boundary for an adversarial attack, providing an *analysis* under the constraints of staying on the tangent space of the manifold and of reaching a distortion budget.
- Investigate theoretically and experimentally the quantization impact on the perturbation.
- Achieve at the same speed as I-FGSM [22] (20 iterations) and under the constraint of a quantization, less distortion



(a) PGD_2 (b) C&W (c) DDN (d) BP (this work)

Fig. 1. Adversarial attacks on a binary classifier in two dimensions. The two class regions are shown in red and blue. Contours indicate class probabilities. The objective is to find a point in the red (*adversarial*) region that is at the minimal distance to input $\mathbf{x}$. Gray (black) paths correspond to low (high) distortion budget ϵ for PGD_2 [22] (a, in green) or parameter λ for C&W [8] (b). The simulation is only meant to illustrate basic properties of the methods. In particular, it does not include Adam optimizer [20] for C&W.

than state-of-the-art attacks including DDN, which needs 100 iterations on ImageNet.
- Propose a benchmark fairly comparing distortion constrained and success constrained attacks (see definitions (3) and (2)).

B. Graphical abstract illustrating the attacks.

To better understand how our attack works, Figure 1 illustrates qualitatively a number of existing attacks technically described in Sect. II-B. On this toy 2d classification problem, the class boundary and the path followed by the optimizer starting at input $\mathbf{x}$ can be easily visualized.

PGD_2, an ℓ_2 version of I-FGSM [22], a.k.a. PGD [27], is controlled by a distortion budget ϵ and eventually follows a path on a ball of radius ϵ centered at $\mathbf{x}$ (*cf*. Fig. 1(a)). Section IV shows that testing different ϵ values is an expensive strategy for finding the optimal distortion budget per image.

C&W [8] depends on a parameter λ that controls the balance between distortion and classification loss. A low value leads to failure. A higher value indeed reaches the optimal perturbation, but with oscillations across the class boundary (Fig. 1(b)). Therefore, an expensive line search over λ is performed internally.

DDN [32] increases or decreases distortion on the fly depending on success and while pointing towards the gradient direction (Fig. 1(c)). It arrives quickly near the optimal perturbation but still suffers from oscillations across the boundary.

On the contrary, *boundary projection* (BP), introduced in this work (*cf*. Fig. 1(d)), cares more about quickly reaching the boundary, not necessarily near the optimal solution, and then walks *along the boundary*, staying mostly in the *adversarial* (red) region. It therefore makes steady progress towards the solution rather than going back and forth.

C. Related works

Before developing the state-of-the-art of white-box attacks in Sect. II-B, we point out other related works.

Optimization on manifolds. In the context of deep learning, stochastic gradient descent on Riemanian manifolds has been studied, *e.g*. RSGD [5] and RSVRG [46]. It is usually applied to manifolds whose geometry is known in analytic form, for instance Grassmann [5] or Stiefel manifolds [18]. In most cases, the motivation is to optimize a very complex function like a classification loss on a well-studied manifold, *e.g*. matrix manifold [1]. On the contrary, we optimize a simple quadratic function (the distortion) on a complex manifold not known in analytic form, *i.e*. a level set of the classification loss.

Information Forensics and Security (IFS). The connection between adversarial examples and the field of IFS has been made recently. Paper [31] provides a conceptual link between machine learning and watermarking allowing to transfer attacks and defenses known in one field to another. Paper [33] adapts a steganalyzer to detect adversarial images.Nearest Neighbours Search (NNS) is another classical tool in IFS. Looking at adversarial examples through the lens of NNS provides a theoretical explanation of the vulnerability of networks producing high intrinsic dimensionality features [2]. The study of their neighborhood has been exploited to design adversarial image detectors [10], [7]. The authors of [13] look at this problem under the framework of game theory. The authors of [38] introduce a secret key into the classifier to prevent white-box attacks. This gives an advantage to the defender following the Kerckhoffs principle. Security threats also hold at training time by poisoning the data to conceal backdoors [4].

Our work focuses on forging adversarial examples. Our intention is to fairly assess their power with or without defenses. Inspired by the methodology for assessing watermarking robustness [16], our protocol introduces in Sect. IV-B the *operating characteristic* of an attack. It reveals the trade-off between distortion and probability of success. We also pay attention to real life conditions obvious in watermarking and steganography but overlooked in adversarial literature: adversarial images are quantized.

II. Problem, Background and State of the Art

A. Problem formulation

Preliminaries. Let $\mathcal{X} := \{0, \Delta, \dots, 1-\Delta, 1\}^n$ with $\Delta := 1/(L-1)$ denote the set of grayscale *images* of n pixels quantized to L levels, and let $\hat{\mathcal{X}} := [0,1]^n$ denote the corresponding real-valued images. An image of more than one color channels is treated independently per channel; in this case n stands for the product of pixels and channels. A *classifier* $f : \hat{\mathcal{X}} \to \mathbb{R}^k$ maps an image $\mathbf{x}$ to a vector $f(\mathbf{x}) \in \mathbb{R}_+^c$ representing probabilities per class over c given classes. The parameters of the classifier are not shown here because they remain fixed in

this work. The classifier *prediction* $\pi : \hat{\mathcal{X}} \to [c] := \{1, \dots, c\}$ maps $\mathbf{x}$ to the class label having the maximum probability:

$$\pi(\mathbf{x}) := \arg\max_{k \in [c]} f(\mathbf{x})_k. \tag{1}$$

The prediction is correct if $\pi(\mathbf{x}) = t$, the *true label*.

Problem. Let $\mathbf{x} \in \mathcal{X}$ be a given image with known true label t. An *adversarial example* $\mathbf{y} \in \mathcal{X}$ is an image such that the *distortion* $\|\mathbf{x} - \mathbf{y}\|$ is small and the probability $f(\mathbf{y})_t$ is also small. This problem takes two forms:

1) *Distortion constrained*:

$$\min_{\mathbf{y} \in \mathcal{X}} f(\mathbf{y})_t \quad \text{subject to} \quad \|\mathbf{x} - \mathbf{y}\| \le \epsilon, \tag{2}$$

where ϵ is a given distortion *budget*. The performance is then measured by the *probability of success* $P_{\text{suc}} := \mathbb{P}(\pi(\mathbf{y}) \neq t)$ as a function of ϵ.

2) *Success constrained*:

$$\min_{\mathbf{y} \in \mathcal{X}} \|\mathbf{x} - \mathbf{y}\| \quad \text{subject to} \quad \pi(\mathbf{y}) \neq t. \tag{3}$$

The performance is then measured by the *expected distortion* $\overline{D} := \mathbb{E}(\|\mathbf{x} - \mathbf{y}\|)$.

This work focuses on the second form, but we present example attacks of both forms in section II-B.

Untargeted attack. The constraint $\pi(\mathbf{y}) \neq t$ in (3) is referred to as an *untargeted* attack, meaning that $\mathbf{y}$ is misclassified regardless of the actual prediction. As an alternative, a *targeted* attack requires that the prediction $\pi(\mathbf{y}) = t'$ is a target label $t' \neq t$. We focus on the former.

Loss function. We focus on a *white-box* attack in this work. Such an attack is specific to f, which is public. In this setting, attacks typically rely on exploiting the gradient of some loss function, using variants of gradient descent. A *classification loss* is defined on the probability vector $\mathbf{p} = f(\mathbf{y})$ with respect to the true label t. For an untargeted attack, this is typically the negative of cross-entropy $\ell(\mathbf{p}, t) := \log p_t$. We should warn that, while the cross-entropy is appropriate for bringing examples into the region of class t during classifier training, its negative is in general *not* appropriate for pulling them out during an attack. This is because this function is mostly flat in the class region. A common solution is to *normalize* the gradient of ℓ [17], [32], assuming it is nonzero. We consider more options in this work. A targeted attack on the other hand may use $-\log p_{t'}$, which works fine because it *brings* examples into class t' region.

Distortion. This work focuses on the 2-norm $\|\cdot\|$ as a measure of distortion. Alternatives like 1-norm and ∞-norm are also common [17], [8]. It is known that none is appropriate for measuring the imperceptibility of adversarial attacks, while more sophisticated measures like structural similarity (SSIM) [43] are limited too [34]. Measuring imperceptibility is arguably as difficult as classification itself.

Integral constraint. The constraint $\mathbf{y} \in \mathcal{X}$ in (2) and (3) is typically relaxed to $\mathbf{y} \in \hat{\mathcal{X}}$ during optimization. Some works conclude the attack by loosely quantizing the optimal solution onto $\mathcal{X}$, typically by *truncation* towards zero. To our knowledge, DDN [32] is the only work to do *rounding* instead, and at the end of each iteration. Quantization is becoming an important issue in adversarial examples because the distortions achieved in recent papers are so small that quantization impacts a lot the perturbations. Appendix A provides a more in-depth study of the impact of the quantization.

B. State-of-the-art Attacks

Distortion constrained attacks. Given a distortion budget ϵ, the *fast gradient sign method* (FGSM) [17] performs a single step in the opposite direction of the (element-wise) sign of the loss gradient with ∞-norm ϵ,

$$\mathbf{y} := \mathbf{x} - \epsilon \operatorname{sign} \nabla_{\mathbf{x}} \ell(f(\mathbf{x}), t). \tag{4}$$

This is the fastest method for problem (2). In the same work adversarial training was introduced, this method quickly generates adversarial examples for training. However, the perturbations are usually high-distortion and visible. The *iterative*-FGSM (I-FGSM) [22] initializes $\mathbf{y}_0 := \mathbf{x}$ and then iterates

$$\mathbf{y}_{i+1} := \operatorname{proj}_{B_\infty[\mathbf{x};\epsilon]}(\mathbf{y}_i - \alpha \operatorname{sign} \nabla_{\mathbf{x}} \ell(f(\mathbf{y}_i), t)), \tag{5}$$

where projection[1] is element-wise to the closed ∞-norm ball $B_\infty[\mathbf{x}; \epsilon]$ of radius ϵ and center $\mathbf{x}$, and also to $\hat{\mathcal{X}}$ (element-wise clipping to interval $[0, 1]$). This method is also known as *basic iterative method* (BIM) [30] and as *projected gradient descent* (PGD) [27]. We refer to as PGD_2 a 2-norm version replacing (5) with

$$\mathbf{y}_{i+1} := \operatorname{proj}_{B_2[\mathbf{x};\epsilon]}(\mathbf{y}_i - \alpha\eta(\nabla_{\mathbf{x}} \ell(f(\mathbf{y}_i), t))), \tag{6}$$

where $\eta(\mathbf{x}) := \mathbf{x}/\|\mathbf{x}\|$ denotes 2-normalization, and projection is to the closed 2-norm ball $B_2[\mathbf{x}; \epsilon]$ of radius ϵ and center $\mathbf{x}$, followed again by element-wise clipping to $[0, 1]$. Although this method is part of Cleverhans library [30], it is not published according to our knowledge.

Success constrained attacks. This family of attacks is typically more expensive. DeepFool [29] is a popular attack which, at each iteration, estimates the distortion needed to go to any class region $k \neq t$ in order to infer which one is the closest.. Paper [37] propose a Lagrangian formulation of problem (3), minimizing the cost function

$$J(\mathbf{y}, \lambda) := \|\mathbf{x} - \mathbf{y}\|^2 + \lambda \ell(f(\mathbf{y}), t), \tag{7}$$

where variable λ is a Lagrange multiplier. They carry out this optimization by box-constrained L-BFGS.

The attack of [8], denoted by C&W in the sequel, pertains to this approach. A change of variable eliminates the box constraint, replacing $\mathbf{y} \in \mathcal{X}$ by $\sigma(\mathbf{w})$, where $\mathbf{w} \in \mathbb{R}^n$ and σ is the element-wise sigmoid function. The classification loss encourages the logit $\log p_t$ to be less than any other $\log p_k$ for $k \neq t$ by at least margin $m \geq 0$,

$$\ell_m(\mathbf{p}, t) := [\log p_t - \max_{k \neq t} \log p_k + m]_+, \tag{8}$$

where $[\cdot]_+$ denotes the positive part. This function is similar to the multi-class SVM loss by Crammer and Singer [11], where $m = 1$, and, apart from the margin, it is a hard version of negative cross-entropy ℓ where softmax is producing the classifier probabilities. It does not have the problem of being

[1] We define $\operatorname{proj}_A(\mathbf{u}) := \arg\min_{\mathbf{v} \in A} \|\mathbf{u} - \mathbf{v}\|$.

flat in the region of class t. The C&W attack uses the Adam optimizer [20] to minimize the cost function

$$J(\mathbf{w}, \lambda) := \|\sigma(\mathbf{w}) - \mathbf{x}\|^2 + \lambda \ell_m(f(\sigma(\mathbf{w})), t). \quad (9)$$

for $\mathbf{w} \in \mathbb{R}^n$. When the margin is reached, loss ℓ_m vanishes and the distortion term pulls $\sigma(\mathbf{w})$ back towards $\mathbf{x}$, causing oscillations around the margin. This is repeated for different λ[2] by line search, which is expensive.

Decoupling direction and norm (DDN) [32] is iterating similarly to PGD_2 (6),

$$\mathbf{y}_{i+1} := \text{proj}_{S[\mathbf{x};\rho_i]}(\mathbf{y}_i - \alpha\eta(\nabla_{\mathbf{x}}\ell(f(\mathbf{y}_i), t))), \quad (10)$$

but projection is to the sphere $S[\mathbf{x}; \rho_i]$ of radius ρ_i and center $\mathbf{x}$, and the radius is adapted to the current distortion: It is set to $\rho_i = (1-\gamma)\|\mathbf{y}_i - \mathbf{x}\|$ if $\mathbf{y}_i$ is adversarial and to $(1+\gamma)\|\mathbf{y}_i - \mathbf{x}\|$ otherwise, where parameter $\gamma \in (0, 1)$. Another difference is that each iteration is concluded by a projection onto $\mathcal{X}$ (rather than $\hat{\mathcal{X}}$) by element-wise clipping to $[0, 1]$ and *rounding*.

Discussion. Optimizing around the class boundary is not a new idea. All of the above attacks do so in order to minimize distortion; implicitly, even distortion constrained attacks like PGD_2 do so, if the minimum parameter ϵ is sought (*cf*. Figure 1(a) and Section IV-B). Even *black-box* attacks do so [6], without having access to the gradient function. The difference of our attack is that our updates are *along* the class boundary, *i.e*., in a direction normal to the gradient.

Other attacks. Generative models like AdvGAN [44] can not be classified as success or distortion constrained attacks. This approach is a tour de force generating adversarial image impressively fast at testing. Yet, the runtime for training is not mentioned and this is an issue because each target class needs a dedicated generative network. The distortion is integrated in the loss at training, but it is not optimized sample by sample since there is no way to control it during testing. Indeed, the perturbation is highly visible in [44, Fig. 3] and has larger distortion than C&W (see their Table 8).

III. Method

Our attack is an *iterative* process with a fixed number K of iterations. Stage 1 aims at quickly producing an adversarial image, whereas Stage 2 is a refinement phase decreasing distortion. The key property of our method is that while in the adversarial region during refinement, it tries to walk along the classification boundary by projecting the distortion gradient onto the tangent hyperplane of the boundary. Hence we call it *boundary projection* (BP).

A. Stage 1

This stage begins at $\mathbf{y}_0 = \mathbf{x}$ and *iteratively* updates in the direction of the gradient of the loss function as summarized in Algorithm 1. The gradient is 2-normalized (line 3) and then scaled by two parameters (line 4): a fixed parameter $\alpha > 0$ and a parameter γ_i that is increasing linearly with iteration $i \leq K$ as follows

$$\gamma_i := \gamma_{\min} + \frac{i}{K+1}(1 - \gamma_{\min}) < 1, \quad (11)$$

[2]Referred to as c in [8].

where $\gamma_{\min} \in (0, \gamma_{\max})$. This makes the updates slow at the beginning to keep distortion low, then faster until the attack succeeds. Parameter α is set empirically to a value large enough so that Stage 1 returns an adversarial image in less than K iterations with high probability. This schedule of γ_i is meant to adjust the level of distortion to each original image, since a single value would be hard to fit all cases. After the update, Clipping is element-wise (line 4).

B. Stage 2

Once Stage 1 has succeeded, Stage 2 continues by *iteratively* considering two cases: if $\mathbf{y}_i$ is adversarial, case OUT aims at minimizing distortion while staying in the adversarial region. Otherwise, case IN aims at decreasing the loss while controlling the distortion. Both work with a first order approximation of the loss around $\mathbf{y}_i$:

$$\ell(f(\mathbf{y}_i + \mathbf{u}), t) \approx \ell(f(\mathbf{y}_i), t) + \mathbf{u}^\top \mathbf{g}, \quad (12)$$

where $\mathbf{g} = \nabla_{\mathbf{x}}\ell(f(\mathbf{y}_i), t)$. The perturbation at iteration i is $\boldsymbol{\delta}_i := \mathbf{y}_i - \mathbf{x}$. Stage 2 is summarized in Algorithm 2. Cases OUT and IN illustrated in Fig. 2 are explained below.

Case OUT takes as input $\mathbf{y}_i$ *outside* class t region, *i.e*. $\pi(\mathbf{y}_i) \neq t$ (line 5). It outputs $\mathbf{y}_{i+1}$ which is a quantized version of vector $\mathbf{z}$ (line 9). The construction of $\mathbf{z}$ stems from two constraints. First, we control the distortion of the next perturbation imposing the constraint $\|\mathbf{z} - \mathbf{x}\| = \epsilon$, so that $\mathbf{z}$ will lie on the hypersphere $S[\mathbf{x}; \epsilon]$. This radius uses again the scheduling (11), $\epsilon = \gamma_i \|\boldsymbol{\delta}_i\| < \|\boldsymbol{\delta}_i\|$, such that updates decelerate to convergence once the attack has already succeeded. We also impose that $\mathbf{z}$ lies on

$$P := \{\mathbf{v} \in \mathbb{R}^n : \langle \mathbf{v} - \mathbf{y}_i, \hat{\mathbf{g}} \rangle = 0\}, \quad (13)$$

the tangent hyperplane of the level set of the loss at $\mathbf{y}_i$, normal to $\hat{\mathbf{g}}$. This second constraint aims at maintaining the value of the loss, up to the first order.

Consider the projection $\mathbf{v}^\star := \mathbf{x} + r\hat{\mathbf{g}}$ of $\mathbf{x}$ onto hyperplane P, where $r := \langle \boldsymbol{\delta}_i, \hat{\mathbf{g}} \rangle$. If $r < \epsilon$, the hypersphere intersects the hyperplane as shown in Fig. 2(a), and both constraints are met. From the infinity of points in this intersection, we pick the one closest to $\mathbf{y}_i$:

$$\mathbf{z} = \mathbf{v}^\star + \eta(\mathbf{y}_i - \mathbf{v}^\star)\sqrt{\epsilon^2 - r^2}. \quad (14)$$

If $r \geq \epsilon$, then $S[\mathbf{x}; \epsilon] \cap P = \emptyset$. We prefer to relax the constraint on the distortion, and choose $\mathbf{z} = \mathbf{v}^\star$, which is by definition

Algorithm 1 Stage 1

Input: $\mathbf{x}$: original image to be attacked
Input: t: true label (untargeted)
Output: $\mathbf{y}$ with $\pi(\mathbf{y}) \neq t$ or failure, iteration i

1: Initialize $\mathbf{y}_0 \leftarrow \mathbf{x}$, $i \leftarrow 0$
2: **while** $(\pi(\mathbf{y}_i) = t) \wedge (i < K)$ **do**
3: $\quad \hat{\mathbf{g}} \leftarrow \eta(\nabla_{\mathbf{x}}\ell(f(\mathbf{y}_i), t))$ ▷ η: 2-normalization
4: $\quad \mathbf{y}_{i+1} \leftarrow \text{clip}_{[0,1]}(\mathbf{y}_i - \alpha\gamma_i\hat{\mathbf{g}})$
5: $\quad i \leftarrow i + 1$
6: **end while**

Algorithm 2 Stage 2

Input: t: true label (untargeted), i current iteration number
Input: $\mathbf{y}_i$: current adversarial image, ϵ: distortion budget
Output: $\mathbf{y}_K$

1: **while** $i < K$ **do**
2: $\quad\boldsymbol{\delta}_i \leftarrow \mathbf{y}_i - \mathbf{x}$ ▷ perturbation
3: $\quad\hat{\mathbf{g}} \leftarrow \eta(\nabla_{\mathbf{x}}\ell(f(\mathbf{y}_i), t))$ ▷ direction
4: $\quad r \leftarrow \langle \boldsymbol{\delta}_i, \hat{\mathbf{g}} \rangle$
5: $\quad$**if** $\pi(\mathbf{y}_i) \neq t$ **then** ▷ OUT
6: $\quad\quad\epsilon \leftarrow \gamma_i \|\boldsymbol{\delta}_i\|$ ▷ distortion control
7: $\quad\quad\mathbf{v}^\star \leftarrow \mathbf{x} + r\hat{\mathbf{g}}$
8: $\quad\quad\mathbf{z} \leftarrow \mathbf{v}^\star + \eta(\mathbf{y}_i - \mathbf{v}^\star)\sqrt{[\epsilon^2 - r^2]_+}$
9: $\quad\quad\mathbf{y}_{i+1} \leftarrow Q_{\text{OUT}}(\mathbf{z}, \mathbf{y}_i)$ ▷ quantization (17)
10: $\quad$**else** ▷ IN
11: $\quad\quad\epsilon \leftarrow \|\boldsymbol{\delta}_i\| / \gamma_i$ ▷ distortion control
12: $\quad\quad\mathbf{z} \leftarrow \mathbf{y}_i - \left(r + \sqrt{\epsilon^2 - \|\boldsymbol{\delta}_i\|^2 + r^2}\right)\hat{\mathbf{g}}$
13: $\quad\quad\mathbf{y}_{i+1} \leftarrow Q_{\text{IN}}(\mathbf{z}, \mathbf{y}_i)$ ▷ quantization (20)
14: $\quad$**end if**
15: $\quad i \leftarrow i + 1$
16: **end while**

the closest point of P to $\mathbf{x}$. Note that $\mathbf{v}^\star = \mathbf{y}_i$ if $\boldsymbol{\delta}_i, \hat{\mathbf{g}}$ are collinear.

This is formally summarized as follows:

$$\mathbf{z} := \arg\min_{\mathbf{v}\in V} \|\mathbf{v} - \mathbf{y}_i\| \tag{15}$$

$$V := \arg\min_{\mathbf{v}\in P} \left| \|\mathbf{v} - \mathbf{x}\| - \epsilon \right|. \tag{16}$$

Here, V is the set of points on P having distortion close to ϵ. If $r < \epsilon$, $V = S[\mathbf{x}; \epsilon] \cap P \neq \emptyset$ as illustrated in Fig. 2(a), otherwise $V = \{\mathbf{v}^\star\}$. The solution $\mathbf{z}$ has a unique closed form, implemented in line 8.

Directly quantizing vector $\mathbf{z}$ onto $\mathcal{X}$ by $Q(\cdot)$, the component-wise rounding, modifies its norm (see App. A). This pulls down our effort to control the distortion. Instead, the process $Q_{\text{OUT}}(\mathbf{z}, \mathbf{y}_i)$ in line 9 looks for the scale β_{OUT} of the perturbation to be applied such that $\|Q(\mathbf{y}_i + \beta(\mathbf{z} - \mathbf{y}_i))\| = \|\mathbf{z}\|$. This is done with a simple line search over β. Then,

$$Q_{\text{OUT}}(\mathbf{z}, \mathbf{y}_i) := Q(\mathbf{y}_i + \beta_{\text{OUT}}(\mathbf{z} - \mathbf{y}_i)). \tag{17}$$

Case IN takes as input $\mathbf{y}_i$ *inside* class t region, *i.e.* $\pi(\mathbf{y}_i) = t$ (line 10). We control distortion $\epsilon = \|\boldsymbol{\delta}_i\| / \gamma_i > \|\boldsymbol{\delta}_i\|$ (11) such that updates decelerate as in Case OUT. We then solve the problem:

$$\mathbf{z} := \arg\min_{\mathbf{v}\in S[\mathbf{x};\epsilon]} \langle \mathbf{v}, \hat{\mathbf{g}} \rangle, \tag{18}$$

i.e., find the point $\mathbf{z}$ at the intersection of sphere $S[\mathbf{x}; \epsilon]$ and the ray through $\mathbf{y}_i$ in the direction opposite of $\mathbf{g}$ as shown in Fig. 2(b). The solution is simple:

$$\mathbf{z} = \mathbf{y}_i - \left(r + \sqrt{\epsilon^2 - \|\boldsymbol{\delta}_i\|^2 + r^2}\right)\hat{\mathbf{g}}, \tag{19}$$

Vector $\mathbf{z}$ moves away from $\mathbf{y}_i$ along direction $-\hat{\mathbf{g}}$ by a step size so to reach $S[\mathbf{x}, \epsilon]$. Case IN is not guaranteed to succeed, but invoking it means that Stage 1 has succeeded.

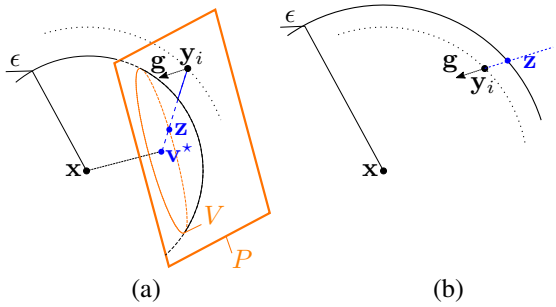


Fig. 2. Refinement stage of BP. Case OUT when $|V| > 1$ (a); case IN (b).

Again a direct rounding jeopardizes the norm of the update $\mathbf{z} - \mathbf{y}_i$. Especially, quantization likely results in $Q(\mathbf{z}) = Q(\mathbf{y}_i)$ if $\|\mathbf{z} - \mathbf{y}_i\| < \beta_{\min} = 0.1$ (see App. A). Instead of a line search as in method OUT, line 13 just makes sure that this event will not happen:

$$Q_{\text{IN}}(\mathbf{z}, \mathbf{y}_i) = Q(\mathbf{y}_i + \beta_{\text{IN}}(\mathbf{z} - \mathbf{y}_i)), \tag{20}$$

with $\beta_{\text{IN}} = \max(1, \beta_{\min} / \|\mathbf{z} - \mathbf{y}_i\|)$.

C. Discussion

The heuristic scheduling (11) builds on a simpler idea of DDN [32], where parameter γ is constant across iterations. This scheduling controls the distortion: In stage 1, updates are small at the beginning to keep distortion low, then larger until the attack succeeds. In stage 2, updates are decreasing as γ_i tends to 1. It increases the distortion when the current image is correctly classified (IN) and decreases the distortion when the current image is adversarial (OUT).

The fact that $(\gamma_i)_i$ is strictly increasing shows that, in Stage 2, an IN iteration (distortion grows by $1/\gamma_i$) followed by an OUT iteration (distortion decays by $\gamma_{i+1} < 1$) is indeed equivalent to a milder IN in the sense that the distortion grows $\gamma_{i+1}/\gamma_i > 1$ smaller than $1/\gamma_i$. Similarly, OUT followed by IN is equivalent to a mild OUT in the sense that distortion decays by $\gamma_i/\gamma_{i+1} < 1$. Both cases lead towards the class boundary by a factor that tends to 1. If the algorithm keeps alternating between OUT and IN and we only look at the OUT iterates (remember, all attacks output the successful iterate of least distortion), this is equivalent to strictly decreasing distortion. This behavior is more stable than having a constant parameter as in DDN.

From all the possible increasing sequences that go to 1 as i goes to the maximum number of iterations, we pick the simplest one: a linear sequence. All this behaviour is controlled by a single parameter, which simplifies the algorithm. That is the only heuristic.

IV. Experiments

In this section we compare our method *boundary projection* (BP) to the attacks presented in Sect. II, namely: FGSM [17], I-FGSM [22], PGD_2 (6), C&W [8], and DDN [32]. This benchmark is carried out on three well-known datasets, with a different neural network for each.

A. Parameters of the attacks

Below we specify the attacks parameters for each dataset. Appendix C details the networks and training parameters.

For the distortion constrained attacks *i.e.* FGSM, I-FGSM and PGD_2, we test a set of ϵ and calculate P_{suc} and $\overline{D}$ according to our evaluation protocol (*cf*. section IV-B). For C&W, we test several parameter settings and pick up the optimum setting as specified below. For DDN, the parameter settings are the default [32], *i.e.* $\epsilon_0 = 1.0$ and $\gamma = 0.05$. **MNIST [24].** App. C details the network with accuracy 0.99. *Parameters*. $\alpha = 0.08$ for I-FGSM, $\alpha = \epsilon/2$ for PGD_2. For C&W: for 5×20 iterations[3], learning rate $\eta = 0.5$ and initial constant $\lambda = 1.0$; for 1×100 iterations, $\eta = 0.1$ and $\lambda = 10.0$. **CIFAR10 [21].** App. C details the network with accuracy 0.93. *Parameters*. $\alpha = 0.08$ for I-FGSM, $\alpha = \epsilon/2$ for PGD_2. For C&W: for 5×20 iterations, learning rate $\eta = 0.1$ and initial constant $\lambda = 0.1$; for 1×100 iterations, $\eta = 0.01$, and $\lambda = 1.0$. **ImageNet [23]** comprises 1,000 images from ImageNet [12]. We use InceptionV3 pre-trained [36] whose accuracy is 0.96. *Parameters*. $\alpha = 0.08$ for I-FGSM, $\alpha = 3$ for PGD_2. For C&W: for 5×20 iterations, learning rate $\eta = 0.01$ and initial constant $\lambda = 20$; for 1×100 iterations, $\eta = 0.01$ and $\lambda = 1.0$.

B. Evaluation protocol

We evaluate an attack by its runtime, two global statistics P_{suc} and $\overline{D}$, and by an operating characteristic curve $D \to \mathsf{P}(D)$ measuring distortion *vs*. probability of success as described below.

Since we focus on the *speed-distortion trade-off*, we measure the required time for all attacks. For the iterative attacks, the complexity of one iteration is largely dominated by the computation of the gradient, which requires one forward and one backward pass through the network. It is thus fair to gauge their complexity by this number, referred to as *iterations* or '# Grads'. Indeed, the actual timings of 100 iterations of I-FGSM, PGD_2, C&W, DDN and BP are 1.08, 1.36, 1.53, 1.46 and 1.17 s/image on average respectively on ImageNet, using Tensorflow, Cleverhans implementation for I-FGSM and C&W, and authors' implementation for DDN.

We measure distortion when *the adversarial images are quantized* by rounding each element to the nearest element in $\mathcal{X}$. This makes sense since adversarial images are meant to be stored or communicated as images rather than real-valued matrices. DDN and BP adversarial images are already quantized. For reference, we report distortion without quantization in Appendix IV-D2.

Given a test set of N' images, we only consider its subset X of N images that are classified correctly without attack. The accuracy of the classifier is N/N'. Let X_{suc} be the subset of X with $N_{\text{suc}} := |X_{\text{suc}}|$ where the attack succeeds and let $D(\mathbf{x}) := \|\mathbf{x} - \mathbf{y}\|$ be the distortion for image $\mathbf{x} \in X_{\text{suc}}$. The global statistics are the *success probability* P_{suc} and *conditional average distortion* $\overline{D}$

$$P_{\text{suc}} := \frac{N_{\text{suc}}}{N}, \quad \overline{D} := \frac{1}{N_{\text{suc}}} \sum_{\mathbf{x} \in X_{\text{suc}}} D(\mathbf{x}). \tag{21}$$

[3]C&W performs line search on λ: "5×20" means 5 values of λ, 20 iterations for each.

TABLE I
SUCCESS PROBABILITY P_{SUC} AND AVERAGE DISTORTION $\overline{D}$ OF OUR METHOD BP ON IMAGENET WITH DIFFERENT *quantization strategies*.

	# Grads	P_{suc}	$\overline{D}$
Rounding in the end	20	1.00	1.44
	100	1.00	1.43
Rounding at each iteration	20	1.00	0.41
	100	1.00	0.32
Rounding with Q_{IN}, Q_{OUT}	20	1.00	0.35
	100	1.00	0.28

Here, $\overline{D}$ is conditioned on success. Indeed, distortion makes no sense for a failure.

We define the *operating characteristic* of a given attack over the set X as the function $\mathsf{P} : [0, D_{\max}] \to [0, 1]$, where $D_{\max} := \max_{\mathbf{x} \in X_{\text{suc}}} D(\mathbf{x})$. Given $D \in [0, D_{\max}]$, $\mathsf{P}(D)$ is the probability of success subject to distortion being upper bounded by D,

$$\mathsf{P}(D) := \frac{1}{N} |\{\mathbf{x} \in X_{\text{suc}} : D(\mathbf{x}) \leq D\}|. \tag{22}$$

This function increases from $\mathsf{P}(0) = 0$ to $\mathsf{P}(D_{\max}) = P_{\text{suc}}$. We sample one intermediate point: $P_{\text{upp}} := \mathsf{P}(D_{\text{upp}})$ is the success rate within a distortion upper bounded by $D_{\text{upp}} \in (0, D_{\max})$.

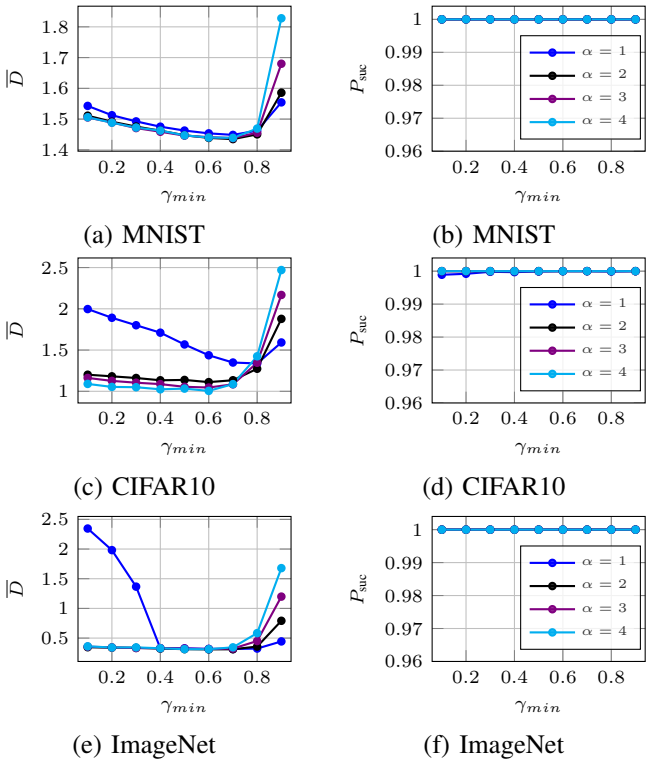
It is difficult to define a fair comparison of *distortion constrained* attacks to *success constrained* attacks (see section II-B). For the first family, we run a given attack several times over the test set with different distortion budget ϵ. The attack succeeds on image $\mathbf{x} \in X$ if it succeeds on at least one of the runs, and the distortion $D(\mathbf{x})$ is the minimum distortion over all successful runs. All statistics are then evaluated as above.

C. Experimental investigations

Before addressing the benchmark, this section investigates on the role of quantization and of the parameters in BP.

1) Quantization: Table I shows the critical role of quantization in our method BP. Since this attack is iterative and works with continuous vectors, one may quantize only at the end of the process, or at the end of each iteration. Another option is to anticipate the detrimental action of quantizing by adapting the length of each step accordingly, as done by $Q_{\text{IN}}(\cdot)$ and $Q_{\text{OUT}}(\cdot)$ in Algorithm 2. The experimental results show that the key is to quantize often so to let the next iterations compensate. Anticipating and adapting gives a substantial extra improvement.

2) Parameter Study: There are two parameters in BP: α and γ_{min}. Both determine the step size of stage 1, while γ_{min} also determines the step size of stage 2. We consider 4 values for α, *i.e.* $1, 2, 3, 4$ and 9 values for γ_{min}, *i.e.* $0.1, 0.2, ..., 0.9$. For each pair of values, we evaluate BP with 20 iterations on a validation set, which we define as a random subset sampled of the training set: 10000 images for MNIST and CIFAR10, and 1000 images for ImageNet. As shown in Fig. 3, success probability is close to one in all cases, while average distortion is in general stable up to $\gamma_{\min} = 0.8$. We choose $\alpha = 2$ and $\gamma_{\min} = 0.7$ for all experiments.



(a) MNIST (b) MNIST

(c) CIFAR10 (d) CIFAR10

(e) ImageNet (f) ImageNet

Fig. 3. Success probability P_{suc} and average distortion $\overline{D}$ for different values of parameters α and γ_{min} of BP with 20 iterations.

D. Benchmark

This section compares different attacks mentioned in this paper, with or without quantization, on various classifiers.

1) Attack evaluation with quantization: Table II summarizes the global statistics of the benchmark. Fig. 4 offers a more detailed view per dataset with operating characteristic.

In terms of average distortion, all iterative attacks perform much better than the single-step FGSM. The performances of C&W are on par with those of I-FGSM, which is unexpected for this more elaborated attack design. The reason is that C&W is put under stress in our benchmark. It usually requires a bigger number of iterations to deliver high quality images. Note that it is possible to avoid the line search on parameter λ as shown in row 1×100. However, it requires a fine tuning so that this single value works over all the images of the dataset. This is not possible for ImageNet.

DDN and our method BP are clearly ahead of the benchmark. DDN yields lower distortion on MNIST at fewer iterations, but its probability of success is not satisfying. DDN is indeed better than BP only on CIFAR10 at 100 iterations. Fig. 4 reveals that the two attacks have similar operating characteristic on all datasets but this is because it refers to 100 iterations.

In terms of success rate, FGSM fails on MNIST; on CIFAR10, I-FGSM and PGD_2 fail as well; finally on ImageNet, C&W fails too. DDN also fails on ImageNet at 20 iterations.

Increasing the number of iterations helps but not at the same rate for all the attacks. For instance, going from 20 to 100 iterations is waste of time for I-FGSM while it is essential for decreasing the distortion of DDN or making PGD_2 efficient on ImageNet. Most importantly, our attack BP brings a dramatic improvement in the speed *vs*. distortion trade-off. Just within 20 iterations, the distortion achieved on ImageNet is very low compared to the others. Section IV-D4 shows the speed *vs*. distortion trade-off in more detail.

Statistics of BP stages are as follows: On CIFAR-10 and MNIST, Stage 1 takes 7 iterations on average. On ImageNet, Stage 1 takes on average 3 iterations out of 20, or 8 iterations out of 100. Appendix B shows examples of images along with corresponding adversarial examples and perturbations for different methods.

2) Attack evaluation without quantization: Table III is the equivalent of Table II but without the integral constraint: the attack is free to output any real matrix provided that the pixel values all belong to $[0, 1]$. When the distortion is large, there is almost no difference.

When an attack delivers low distortion on average with real matrices, the quantization may lower the probability of success. This is especially true with the iterative attacks finding adversarial examples just nearby the border between the two classes. Quantization jeopardizes this point and sometimes brings it back in the true class region. More importantly, the impact of the quantization on the distortion is no longer negligible. This is clearly visible when comparing Table III and Table II for DDN and BP over ImageNet.

Similarly, Fig. 5 is the equivalent of Fig. 4 without the integral constraint. By comparing the two figures, it can be seen that PGD_2 and C&W, but also DDN and BP, are improving on ImageNet by having significantly lower distortion. This agrees with measurements of success rate in Table III, where PGD_2 and C&W are not failing as they do in Table II with quantization. Our BP is still the strongest attack over all datasets.

3) Attack evaluation on robust models: Table IV is similar to Table II but is evaluating attacks on robust models. In particular, on MNIST and CIFAR10, we use the same models as described in Section IV-A, which we adversarially train according to [27]. On ImageNet, we use off-the shelf[4] InceptionV3 obtained by *ensemble adversarial training* on four models [40].

In general, DDN and BP outperform all other attacks in terms of either average distortion $\overline{D}$ or success rate P_{upp}. On ImageNet in particular, all other attacks have significantly higher distortion and fail in terms of success rate. DDN has significantly greater distortion than BP and fails in terms of success rate at 20 iterations, while at 100 iterations BP still has lower distortion. DDN and BP have similar performance on CIFAR10. On MNIST, DDN fails in terms of probability of success at 20 iterations, while at 100 iterations BP is superior.

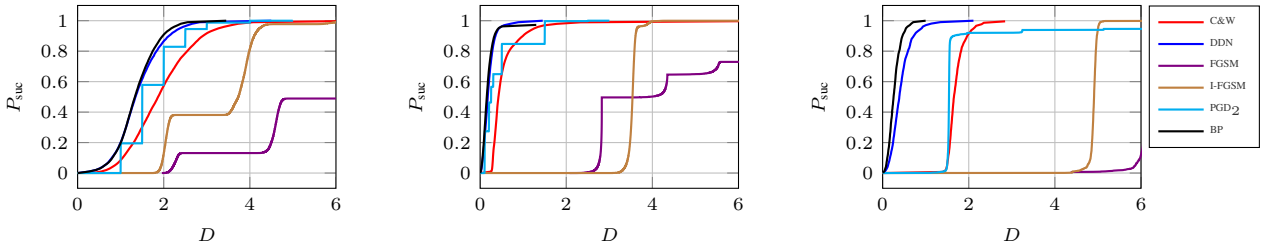
Fig. 6 shows a more detailed view of operating characteristics, similarly to Fig. 4 for models trained on natural images. We can see that BP is still ahead of the competition. It is close to DDN, but this is because Fig. 6 refers to 100 iterations. The two attacks outperform all others by a large margin.

4) Speed vs. *distortion trade-off:* Figure 7(a) is a graphical view of some results reported in Table II with more choices

[4]https://github.com/tensorflow/models/tree/master/research/adv_imagenet_models

TABLE II
SUCCESS PROBABILITY P_{SUC} AND AVERAGE DISTORTION $\overline{D}$ WITH QUANTIZATION. P_{UPP} IS THE SUCCESS RATE UNDER DISTORTION BUDGET $D_{\text{UPP}} = 2$ FOR MNIST, 0.7 FOR CIFAR10, AND 1 FOR IMAGENET.

		MNIST			CIFAR10			ImageNet		
Attack	# Grads	P_{suc}	$\overline{D}$	P_{upp}	P_{suc}	$\overline{D}$	P_{upp}	P_{suc}	$\overline{D}$	P_{upp}
FGSM	1	0.99	5.80	0.00	0.95	5.65	0.00	0.88	9.18	0.00
I-FGSM	20	1.00	3.29	0.17	1.00	3.54	0.00	1.00	4.90	0.00
	100	1.00	3.23	0.18	1.00	3.53	0.00	1.00	4.90	0.00
PGD_2	20	1.00	1.80	0.63	1.00	0.66	0.76	0.63	3.63	0.00
	100	1.00	1.74	0.66	1.00	0.60	0.84	1.00	1.85	0.00
C&W	5×20	1.00	1.94	0.56	0.99	0.56	0.81	1.00	1.70	0.00
	1×100	0.98	1.90	0.57	0.87	0.38	0.76	0.97	2.57	0.00
DDN	20	0.82	1.40	0.70	1.00	0.63	0.74	0.99	1.18	0.05
	100	1.00	1.41	0.87	1.00	0.21	0.98	1.00	0.43	0.97
BP (this work)	20	1.00	1.45	0.86	0.97	0.49	0.87	1.00	0.35	0.96
	100	1.00	1.37	0.91	0.97	0.30	0.97	1.00	0.28	1.00



(a) MNIST (b) CIFAR10 (c) ImageNet

Fig. 4. Operating characteristics on MNIST, CIFAR10 and ImageNet. The number of iterations is 5×20 for C&W and 100 for I-FGSM, PGD_2, DDN and our BP.

TABLE III
SUCCESS PROBABILITY P_{SUC} AND AVERAGE DISTORTION $\overline{D}$ *without quantization*. P_{UPP} MEASURED AT $D_{\text{UPP}} = 2$ FOR MNIST, 0.7 FOR CIFAR10, AND 1 FOR IMAGENET.

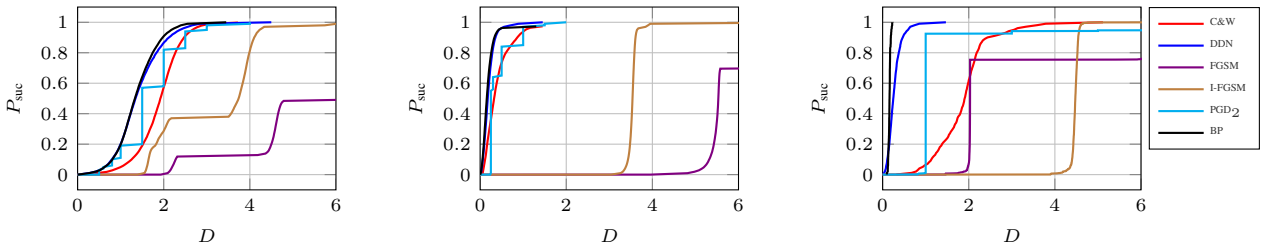
		MNIST			CIFAR10			ImageNet		
Attack	# Grads	P_{suc}	$\overline{D}$	P_{upp}	P_{suc}	$\overline{D}$	P_{upp}	P_{suc}	$\overline{D}$	P_{upp}
FGSM	1	0.99	5.81	0.00	0.97	4.78	0.00	0.85	3.02	0.00
I-FGSM	20	1.00	3.22	0.27	1.00	3.54	0.00	1.00	4.47	0.00
	100	1.00	3.16	0.29	1.00	3.53	0.00	1.00	4.47	0.00
PGD_2	20	1.00	1.76	0.63	1.00	0.51	0.77	0.64	3.94	0.36
	100	1.00	1.70	0.66	1.00	0.43	0.85	0.95	1.11	0.61
C&W	5×20	1.00	1.93	0.56	1.00	0.56	0.81	1.00	1.37	0.23
	1×100	1.00	1.89	0.57	0.97	0.38	0.84	1.00	1.87	0.06
DDN	20	0.82	1.39	0.70	1.00	0.62	0.74	1.00	0.76	0.95
	100	1.00	1.41	0.87	1.00	0.20	0.98	1.00	0.28	0.99
BP (this work)	20	1.00	1.41	0.86	0.97	0.33	0.87	1.00	0.20	1.00
	100	1.00	1.35	0.91	0.97	0.18	0.97	1.00	0.16	1.00

of number of iterations between 20 and 100, and only for ImageNet where our performance gain is the most significant. Just within 20 iterations, its distortion $\overline{D}$ is already so much lower than that of other attacks, that its decrease (-20% at 100 iterations) is not visible in Fig. 7. On the contrary, more iterations are useless for I-FGSM, and PGD_2 achieves low distortion only with more than 50 iterations. Figure 7(b) confirms that the probability of success is close to 1 for both DDN and BP for the numbers of iterations considered.

E. Defense evaluation with adversarial training

We also test under adversarial training [17]. The network is re-trained with a dataset composed of the original training set and the corresponding adversarial images. This training is special: at the end of each epoch, the network is updated and fixed, then the adversarial images for this new update are forged by some reference attack, and the next epoch starts with this new set. This is tractable only if the reference attack is fast. We use it with FGSM as the reference attack.

It is more interesting to study DDN and BP as alternatives to FGSM: at 20 iterations, they are fast enough to play the role of the reference attack in adversarial training. In this case, we follow the training process suggested by [32]: the model is first trained on clean examples, then fine-tuned for 30 iterations with adversarial examples. As shown in Table V, DDN and BP perform equally better than FGSM on CIFAR10, in terms of either average distortion or success rate. Among the reliable attacks (*i.e.* whose P_{suc} is close to 1), the worst attack now requires a distortion three times larger than the distortion of the worst attack without defense. In the same way, on MNIST,

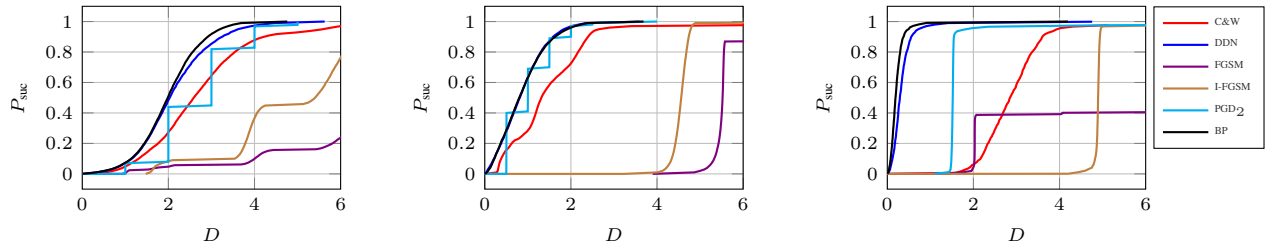


(a) MNIST (b) CIFAR10 (c) ImageNet

Fig. 5. Operating characteristics on MNIST, CIFAR10 and ImageNet *without quantization*. The number of iterations is 5×20 for C&W and 100 for I-FGSM, PGD$_2$, DDN and our BP.

TABLE IV
SUCCESS PROBABILITY P_{SUC}, AVERAGE DISTORTION $\overline{D}$, AND SUCCESS RATE P_{UPP} UNDER *adversarial training* WITH PGD AS THE REFERENCE ATTACK, FOLLOWING [27] FOR MNIST AND CIFAR10; AND *ensemble adversarial training* [40] FOR IMAGENET. P_{UPP} MEASURED AT $D_{\text{UPP}} = 2$ FOR MNIST, 0.7 FOR CIFAR10, AND 1 FOR IMAGENET.

		MNIST [27]			CIFAR10 [27]			ImageNet [40]		
Attack	# Grads	P_{suc}	$\overline{D}$	P_{upp}	P_{suc}	$\overline{D}$	P_{upp}	P_{suc}	$\overline{D}$	P_{upp}
FGSM	1	0.48	5.69	0.05	0.98	6.21	0.00	0.44	2.98	0.00
I-FGSM	20	1.00	4.99	0.08	1.00	4.53	0.00	1.00	4.92	0.00
	100	1.00	4.99	0.08	1.00	4.56	0.00	1.00	4.93	0.00
PGD$_2$	20	0.99	2.76	0.19	1.00	1.03	0.41	0.76	2.14	0.00
	100	1.00	2.68	0.20	1.00	1.02	0.41	0.98	1.59	0.00
C&W	5×20	0.99	2.75	0.27	0.98	1.41	0.22	0.98	2.85	0.00
	1×100	0.94	2.22	0.34	0.60	0.77	0.27	0.97	2.41	0.00
DDN	20	0.43	1.61	0.32	0.97	0.92	0.41	0.99	1.10	0.23
	100	1.00	2.12	0.48	1.00	0.87	0.42	1.00	0.34	0.98
BP (this work)	20	1.00	2.17	0.46	1.00	0.94	0.41	1.00	0.35	0.94
	100	1.00	2.00	0.51	1.00	0.88	0.43	1.00	0.23	0.99



(a) MNIST (b) CIFAR10 (c) ImageNet ens4

Fig. 6. Operating characteristics of attacks against *robust models*: *adversarial training* with PGD as the reference attack [27] for MNIST and CIFAR10, and *ensemble* adversarial training [40] for ImageNet. The number of iterations is 5×20 for C&W and 100 for I-FGSM, PGD$_2$, DDN and our BP.

the distortion of the worst case attack doubles going from 1.37 (baseline) to 2.73 (BP defense). In most cases, BP is a better defense than DDN, forcing the attacker to have 20% more distortion. Note that for a given defense, the strongest attack is almost always BP.

V. CONCLUSION

The main idea of BP is to travel on the manifold defined by the class boundary while seeking to minimize distortion. This travel is operated by the refinement stage, which alternates on both sides of the boundary, but attempts to stay mostly in the adversarial region. Referring to section II-A, BP is in effect doing for the *success constrained* problem what PGD$_2$ is doing for the *distortion constrained* problem: BP minimizes distortion on the class boundary manifold (a level set of the classification loss), while PGD$_2$ minimizes the classification loss on a sphere (a level set of the distortion).

BP also takes into account the detrimental effect of *quantization*. By doing so, the amplitude of the perturbation is controlled from one iteration to another. The main advantage of our attack is the small number of iterations required to achieve both reliability (probability of success close to one) and high quality (low average distortion).

APPENDIX A
PREDICTING DISTORTION AFTER QUANTIZATION

This appendix aims at predicting when the quantization cancels the perturbation, assuming that they are independent of each other. Iteration i starts with a quantized image $\mathbf{y}_i \in \mathcal{X}$, adds update $\mathbf{u} \in \mathbb{R}^n$, and then quantizes s.t. $\mathbf{y}_{i+1} = Q(\mathbf{y}_i + \mathbf{u})$. Quantization is done by rounding with step $\Delta := 1/(L-1)$: $y_{i+1,j} = y_{i,j} + e_j$ if $u_j \in (e_j - \Delta/2, e_j + \Delta/2]$ with $e_j \in \Delta\mathbb{Z}$. Border effects where $y_{i,j} + e_j \notin \mathcal{X}$ are neglected.

TABLE V
SUCCESS PROBABILITY P_{SUC}, AVERAGE DISTORTION $\overline{D}$, AND SUCCESS RATE P_{UPP} UNDER *adversarial training* DEFENSE WITH I-FGSM, DDN, OR BP AS THE REFERENCE ATTACK. P_{UPP} MEASURED AT DISTORTION $D_{\text{UPP}} = 2$ FOR MNIST, AND 0.7 FOR CIFAR10.

		MNIST						CIFAR10					
Attack →		PGD_2		DDN		BP		PGD_2		DDN		BP	
↓ Defense		20	100	20	100	20	100	20	100	20	100	20	100
baseline	P_{suc}	1.00	1.00	0.82	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.97	0.97
	$\overline{D}$	1.80	1.74	1.40	1.41	1.45	1.37	0.66	0.59	0.63	0.21	0.49	0.30
	P_{upp}	0.63	0.66	0.70	0.87	0.86	0.91	0.76	0.84	0.74	0.98	0.87	0.97
FGSM	P_{suc}	1.00	1.00	0.51	1.00	0.89	1.00	1.00	1.00	1.00	1.00	0.99	1.00
	$\overline{D}$	1.92	1.85	1.28	1.60	1.92	1.58	0.68	0.62	0.59	0.24	0.67	0.24
	P_{upp}	0.48	0.53	0.44	0.72	0.53	0.73	0.72	0.79	0.80	0.98	0.72	0.99
DDN	P_{suc}	0.99	1.00	0.29	1.00	0.99	1.00	1.00	1.00	0.98	1.00	1.00	1.00
	$\overline{D}$	3.03	2.89	1.68	2.38	2.69	2.27	0.95	0.94	0.77	0.71	0.75	0.68
	P_{upp}	0.12	0.14	0.20	0.32	0.28	0.34	0.52	0.52	0.54	0.55	0.56	0.58
BP	P_{suc}	0.94	0.96	0.36	1.00	0.95	1.00	1.00	1.00	0.97	1.00	1.00	1.00
	$\overline{D}$	3.14	3.12	1.65	2.81	2.98	2.73	0.96	0.94	0.75	0.70	0.76	0.69
	P_{upp}	0.15	0.15	0.24	0.27	0.25	0.26	0.55	0.55	0.57	0.59	0.56	0.59

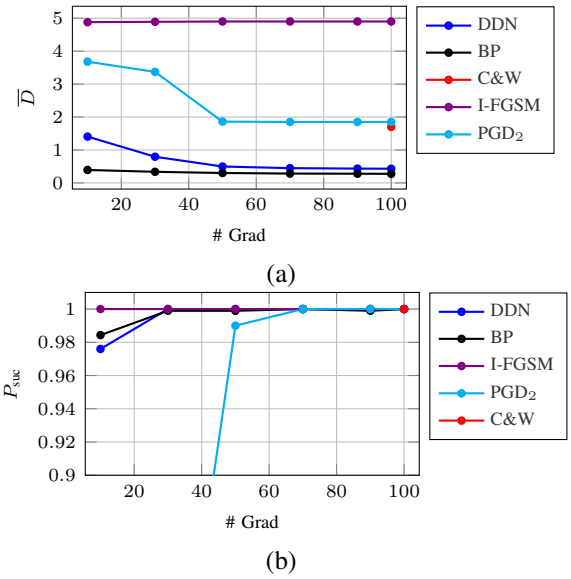


Fig. 7. (a) Average distortion *vs*. number of iterations on ImageNet. I-FGSM is not improving with iterations because it is constrained by ϵ. (b) Corresponding probability of success.

We now take a statistical point of view where the update is modelled by a random vector $\mathbf{U}$ uniformly distributed over the hypersphere of radius ρ, the norm of the perturbation before quantization. The quantization noise is now random, denoted by $E_j \in \Delta\mathbb{Z}$ for pixel j, introducing a distortion

$$D^2 = \sum_{j=1}^{n} (y_{i+1,j} - y_{i,j})^2 = \sum_{j=1}^{n} E_j^2. \tag{23}$$

The expectation of a sum is always the sum of the expectations, whatever the dependence between the summands: $\mathbb{E}(D^2) = \sum_{j=1}^{n} \mathbb{E}(E_j^2) = n\mathbb{E}(E_j^2)$. This expectation is not null if $\mathbb{P}(|E_j| \geq \Delta) > 0$ since $\mathbb{E}(D^2) \geq n\Delta^2 \mathbb{P}(|E_j| \geq \Delta)$.

This r.v. E_j takes a value depending on the scalar product $S_j := \mathbf{U}^\top \mathbf{c}_j$, where $\mathbf{c}_j$ is the j-th canonical vector. This scalar product lies in $[-\rho, \rho]$, so that $\mathbb{P}(E_j \geq \Delta) = 0$ if $\Delta/2 > \rho$. Otherwise, $|E_j| \geq \Delta$ when $|S_j| \geq \Delta/2$, which happens when $\mathbf{U}$ lies inside the dual hypercone of axis $\mathbf{c}_j$ and semi-angle $\theta = \arccos(c)$ with $c := \Delta/2\rho$. The probability of this event is equal to the ratio of the solid angles of this dual hypercone and the full space $\mathbb{R}^n$. This quantity can be expressed via the incomplete regularized beta function I:

$$\mathbb{P}(|E_j| \geq \Delta) = \begin{cases} 1 - I_{c^2}(1/2, (n-1)/2), & \text{if } c \leq 1 \\ 0, & \text{otherwise} \end{cases}$$

For large n, this probability approximately equals $2\Phi(-\sqrt{n}c)$.

In the end, the lower bound of $\mathbb{E}(D^2)$ after quantization depends on Δ, n, and ρ the norm of the perturbation before quantization. When $n = 3 * 299^2$ (*i.e.* ImageNet), this lower bound equals Δ^2 (*i.e.* the smallest distortion if not null) for $\rho = 0.1$. When the update has a smaller norm, quantization is likely to kill it, $\mathbf{y}_{i+1} = \mathbf{y}_i$, and we waste one iteration.

APPENDIX B
ADVERSARIAL IMAGE EXAMPLES

Fig. 8 shows the worst-case ImageNet examples for BP along with the images generated by all methods and the corresponding normalized perturbations. FGSM has the highest distortion over all methods and BP the lowest. DDN has the highest ∞-norm distortion. Observe that for no method is the perturbation visible, although this is a worst-case example.

APPENDIX C
NETWORKS AND TRAINING PARAMETERS

MNIST [24]. We use a simple network with three convolutional layers and one fully connected layer. The first convolutional layer has 64 features, kernel of size 8 and stride 2; the second has 128 features, kernel 6 and stride 2; the third has 128 features, kernel 5 and stride 1. It uses LeakyRelu activation [26]. The loss function is the cross-entropy.

We train the network with the $60,000$ images of the training set. After a random initialization, training lasts 6 epochs with batch size 128, learning rate 0.001, and the optimizer is Adam. Between epochs, the training data are shuffled.

CIFAR10 [21]. We use a simple CNN network with nine convolutional layers, two max-pooling layers, two dropout layers, ending in global average pooling and a fully connected layer. Batch normalization [19] is applied after every convolutional layer. It also uses LeakyRelu. The loss function is

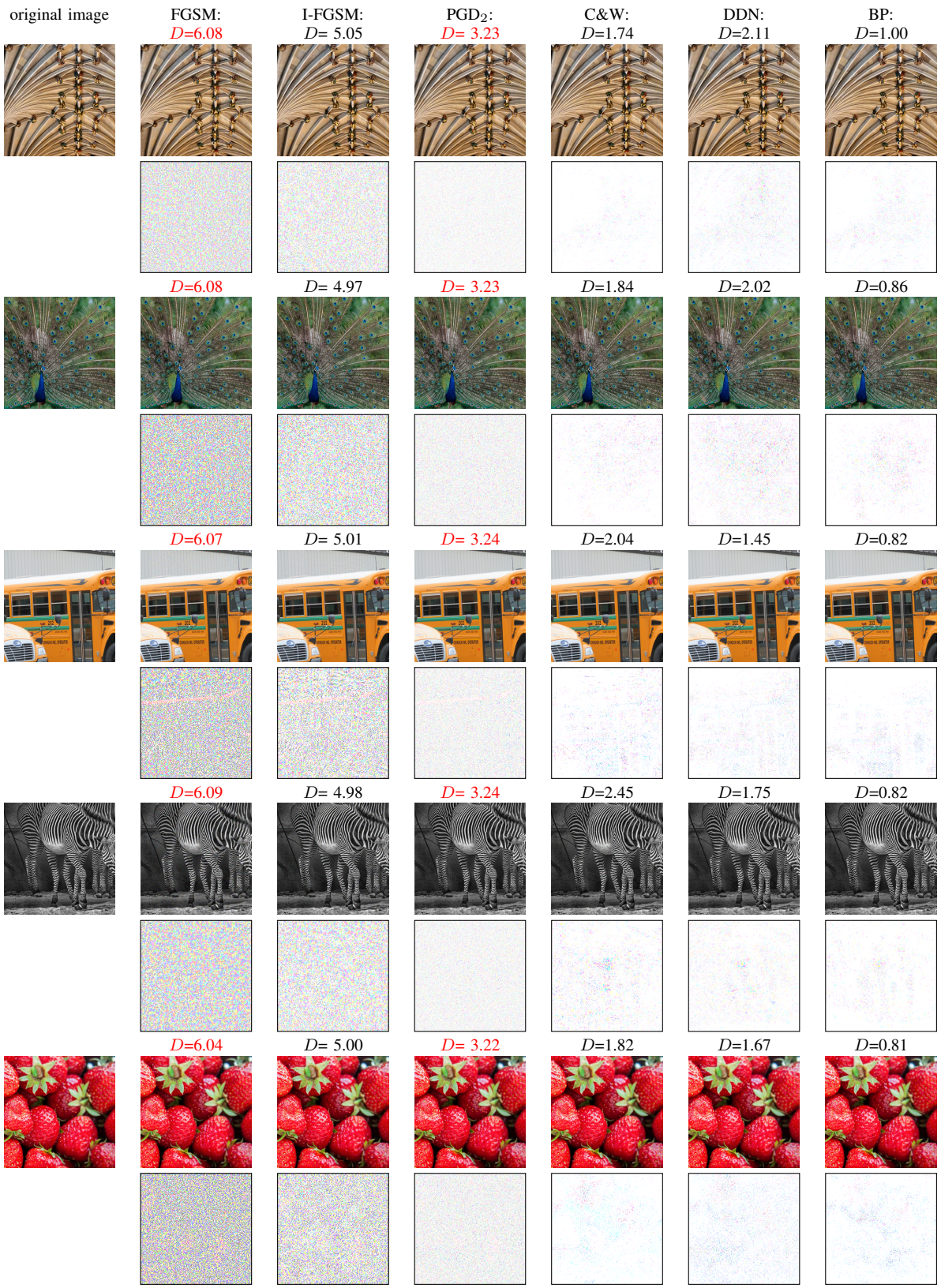


Fig. 8. Original (left), adversarial (top row) and scaled perturbation (below) images against InceptionV3 on ImageNet. The five images are the worst 5 images for BP requiring the strongest distortions, yet these are smaller than the distortions necessary with all other methods (The red color means that the forged image is not adversarial). Perturbations are inverted (low is white; high is colored, per channel) and scaled in the same way for a fair comparison.

the cross-entropy. All the kernels of the convolutional layers are initialized with HeReLuNormal initializer, and their kernel size is 3. For the first three convolutional layers, the number of filters is 128 and the padding mode is 'same'. For the following three convolutional layers, the number of filters is 256 and the padding mode is 'same'. For the last three convolutional layers, the padding mode is 'valid'. The seventh layer has 512 filters, while 256 filters for the eighth layer, and 128 filters for the last layer. The parameter α for LeakyRelu is 0.1, and the rate for dropout is 0.5. The pool size of the max pooling layer is 2, the strides shape is 2 and the padding mode is 'valid'. The pool size of the average pooling layer is 2 and the strides shape is 6 and the padding mode is 'valid'.

We trained the model with the $60,000$ images of the training set. After a random initialization, training lasts 200 epochs with batch size 128, learning rate 0.001, and the optimizer is Adam. Between epochs, the training data are shuffled.

ImageNet [23]. We use InceptionV3 [36] with the pre-trained model from TensorFlow-Slim image classification library[5].

Robust models - Attack evaluation. We use the same network for MNIST and CIFAR10, but the training differs. We follow the adversarial training method [27], *i.e*. the models are trained with training data and their adversarial version generated by PGD_2. For MNIST, PGD_2 iterates 40 times with $\epsilon = 0.3$ while $\alpha = 0.01$. The training batch size is 50 over 100 epochs. For CIFAR10, PGD_2 iterates 100 times, with $\epsilon = 8/255$ while $\alpha = 2/255$. The training batch size is 128 over 200 epochs. Between epochs, the training data are shuffled.

The robust model for ImageNet is ensemble adversarial training [40], which is directly taken from TensorFlow library[6].

Robust models - Defense evaluation. The networks on MNIST and CIFAR10 are the same as before.

On MNIST, it is trained from scratch with the same setup but with training data and their adversarial examples. FGSM defense model uses FGSM with $\epsilon = 0.3$. DDN and BP defense model use these attacks with 20 iterations and the same parameters as described in the attack methods.

On CIFAR10, only FGSM defense model is trained from scratch. For DDN and BP defense model, we follow the training suggested by [32]. The model is first trained on clean examples, then fine-tuned for 30 iterations with adversarial examples. The parameters are initialized randomly. The optimizer is Momentum Optimizer, the initial learning rate is 0.001 and the momentum is 0.9. It is trained with 200 epochs and the batch size 128. Between epochs, the training data are shuffled. FGSM defense model uses FGSM with $\epsilon = 0.3$. DDN and BP defense model use these attacks with 20 iterations and same parameters as attack methods.

Experiments run on TensorFlow1.8.0-py2.7 over CUDA 9.0.176 ; Cleverhans [30] v2.0.0 produces the existing attacks.

ACKNOWLEDGMENT

T.F. is supported by the ANR-AID chaire SAIDA.

[5]https://github.com/tensorflow/models/tree/master/research/slim

[6]https://github.com/tensorflow/models/tree/master/research/adv_imagenet_models

REFERENCES

[1] P.-A. Absil, R. Mahony, and R. Sepulchre. *Optimization algorithms on matrix manifolds*. Princeton University Press, 2009.

[2] L. Amsaleg, J. E. Bailey, D. Barbe, S. Erfani, M. E. Houle, V. Nguyen, and M. Radovanovic. The Vulnerability of Learning to Adversarial Perturbation Increases with Intrinsic Dimensionality. In *Proc. of WIFS 2017*, Rennes, France, December 2017.

[3] A. Azulay and Y. Weiss. Why do deep convolutional networks generalize so poorly to small image transformations? Technical report, 2018.

[4] M. Barni, K. Kallas, and B. Tondi. A new backdoor attack in cnns by training set corruption without label poisoning. In *2019 IEEE International Conference on Image Processing (ICIP)*, pages 101–105, Sep. 2019.

[5] S. Bonnabel. Stochastic gradient descent on riemannian manifolds. *IEEE Transactions on Automatic Control*, 58(9):2217–2229, 2013.

[6] W. Brendel, J. Rauber, and M. Bethge. Decision-based adversarial attacks: Reliable attacks against black-box machine learning models. In *ICLR*, 2018.

[7] R. Caldelli, R. Becarelli, F. Carrara, F. Falchi, and G. Amato. Exploiting cnn layer activations to improve adversarial image classification. In *2019 IEEE International Conference on Image Processing (ICIP)*, pages 2289–2293, Sep. 2019.

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

[9] N. Carlini and D. Wagner. Audio adversarial examples: Targeted attacks on speech-to-text. Technical report, 2018.

[10] F. Carrara, F. Falchi, R. Caldelli, G. Amato, and R. Becarelli. Adversarial image detection in deep neural networks. *Multimedia Tools and Applications*, 78(3):2815–2835, Feb 2019.

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

[12] 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.

[13] L. Dritsoula, P. Loiseau, and J. Musacchio. A game-theoretic analysis of adversarial classification. *IEEE Trans. Information Forensics and Security*, 12(12):3094–3109, 2017.

[14] L. Engstrom, B. Tran, D. Tsipras, L. Schmidt, and A. Madry. A rotation and a translation suffice: Fooling cnns with simple transformations. Technical report, 2017.

[15] A. Fawzi, S.-M. Moosavi-Dezfooli, and P. Frossard. Robustness of classifiers: From adversarial to random noise. Technical report, 2016.

[16] T. Furon and P. Bas. Broken arrows. *EURASIP Journal on Information Security*, 2008(1), 2008.

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

[18] M. Harandi and B. Fernando. Generalized backpropagation, etude de cas: Orthogonality. Technical report, 2016.

[19] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, 2015.

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

[21] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. Technical report, 2009.

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

[23] 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.

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

[25] J. Li, R. Ji, H. Liu, X. Hong, Y. Gao, and Q. Tian. Universal perturbation attack against image retrieval. Technical report, 2018.

[26] A. L. Maas, A. Y. Hannun, and A. Y. Ng. Rectifier nonlinearities improve neural network acoustic models. In *ICML*, volume 30, 2013.

[27] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv:1706.06083*, 2017.

[28] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard. Universal adversarial perturbations. *arXiv:1610.08401*, 2016.

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

[30] 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.
[31] 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.
[32] J. Rony, L. G. Hafemann, L. S. Oliveira, I. B. Ayed, R. Sabourin, and E. Granger. Decoupling direction and norm for efficient gradient-based l2 adversarial attacks and defenses. Technical report, 2018.
[33] P. Schöttle, A. Schlögl, C. Pasquini, and R. Böhme. Detecting adversarial examples - a lesson from multimedia security. In *2018 26th European Signal Processing Conference (EUSIPCO)*, pages 947–951, Sep. 2018.
[34] 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.
[35] C.-J. Simon-Gabriel, Y. Ollivier, B. Schölkopf, L. Bottou, and D. Lopez-Paz. Adversarial vulnerability of neural networks increases with input dimension. Technical report, 2018.
[36] 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.
[37] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. *arXiv:1312.6199*, 2013.
[38] O. Taran, S. Rezaeifar, T. Holotyak, and S. Voloshynovskiy. Defending against adversarial attacks by randomized diversification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 11226–11233, 2019.
[39] G. Tolias, F. Radenovic, and O. Chum. Targeted mismatch adversarial attack: Query with a flower to retrieve the tower. In *Proc. of ICCV*, 2019.
[40] F. Tramèr, A. Kurakin, N. Papernot, D. Boneh, and P. McDaniel. Ensemble adversarial training: Attacks and defenses. *arXiv:1705.07204*, 2017.
[41] F. Tramèr, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel. The space of transferable adversarial examples. *arXiv:1704.03453*, 2017.
[42] Y. Wang, X. Ma, J. Bailey, J. Yi, B. Zhou, and Q. Gu. On the convergence and robustness of adversarial training. In *ICML*, 2019.
[43] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Trans. on image processing*, 13(4), 2004.
[44] C. Xiao, B. Li, J.-Y. Zhu, W. He, M. Liu, and D. Song. Generating adversarial examples with adversarial networks. *arXiv preprint arXiv:1801.02610*, 2018.
[45] C. Xie, J. Wang, Z. Zhang, Y. Zhou, L. Xie, and A. Yuille. Adversarial examples for semantic segmentation and object detection. *arXiv:1703.08603*, 2017.
[46] H. Zhang, S. J. Reddi, and S. Sra. Riemannian svrg: Fast stochastic optimization on riemannian manifolds. In *NIPS*, pages 4592–4600, 2016.

Hanwei Zhang received her bachelor's degree in computer science in 2015 and the M.Sc. degree in 2017 from East China Normal University. She is now a Ph.D student of PRogram Of Sino-French Education for Research (PROSFER) between East China Normal University and cole Normale Suprieure de Rennes. She works at IRISA Rennes under the supervision of Laurent Amsaleg, Yannis Avrithis and Teddy Furon in the Linkmedia team.

During the Ph.D, her research focuses on security issues in machine learning, especially on adversarial examples for deep learning systems.

Yannis Avrithis received the MSc degree in Communications and Signal Processing from the University of London, UK, in 1994, and the PhD degree from National Technical University of Athens, Greece, in 2001.

He is currently a Research Scientist at Inria Rennes-Bretagne Atlantique, carrying out research on computer vision and machine learning. Before that he was at the National and Kapodistrian University of Athens and at the National Technical University of Athens, where he lead the Image and Video Analysis research team.

He has been involved in 16 European, 5 French and 10 Greek research projects. He has co-supervised 12 PhD theses and 16 Diploma theses. He has published 3 theses, 3 edited volumes, 29 articles in journals, 112 in conferences and workshops, and 8 book chapters.

Yannis Avrithis has contributed to the organization of 23 conferences and workshops, and is a Reviewer in 15 scientific journals and 15 conferences. He has been Associate Editor for EURASIP Journal on Image and Video Processing.

Teddy Furon received the M.Sc. degree in 1998 and the PhD degree in signal processing in 2002, both from Telecom ParisTech. His fields of interest is the security related to multimedia, signal processing, and artificial intelligence.

He has worked both in industry (Thomson, Technicolor) and academia (Univ. Cath. de Louvain, Belgium, and now Inria Rennes, France, in the Linkmedia team). He co-founded of the company LAMARK protecting rights of photo agencies.

M. Teddy Furon received the Brittany Best Young Researcher prize in 2006. He is the co-author of 80 conference papers, 20 journal articles, 6 book chapters and 9 patents. He has been Associate Editor for four journals, including IEEE Trans. on Inf. Forensics and Security.

Laurent Amsaleg received his PhD from the University of Paris 6. He is now a senior researcher at CNRS. He leads the Linkmedia research group, at the IRISA/INRIA Lab in Rennes, France. His research interests include high dimensional indexing at scale and multimedia analytics, as well as the many facets of the security issues in relation with the processing of extremely large collections of multimedia material. Topics dealing with privacy and adversarial machine learning are therefore central to his work.