ADef: an Iterative Algorithm to Construct Adversarial Deformations

R. Alaifari and G.S. Alberti and T. Gauksson

Research Report No. 2018-15
April 2018

ADef: an Iterative Algorithm to Construct Adversarial Deformations

Rima Alaifari[1], Giovanni S. Alberti[2], and Tandri Gauksson[1]

[1]Department of Mathematics, ETH Zurich
[2]Department of Mathematics, University of Genoa

Abstract

While deep neural networks have proven to be a powerful tool for many recognition and classification tasks, their stability properties are still not well understood. In the past, image classifiers have been shown to be vulnerable to so-called adversarial attacks, which are created by additively perturbing the correctly classified image.

In this paper, we propose the ADef algorithm to construct a different kind of adversarial attack created by iteratively applying small deformations to the image, found through a gradient descent step. We demonstrate our results on MNIST with a convolutional neural network and on ImageNet with Inception-v3 and ResNet-101.

1 Introduction

In the last years, deep neural networks have outperformed all previous techniques in many machine learning tasks, including image classification. This is made possible by two main ingredients. First, the depth of the network which is in general composed of many layers, and second, the high degree of freedom in choosing the parameters in each affine transformation. However, the advantage brought by these two aspects comes at a price: if the norms of the weight matrices are not small, the application of several layers of operations results in networks that are in general unstable [25]. In other words, two very close signals (e.g., two visually indistinguishable images) may have very different outputs, resulting in one of them being misclassified even if the other one is correctly classified with high confidence.

Since the first observation in [25], a lot of research has been done to investigate this issue through the construction of adversarial examples: given a correctly classified image x, we look for an image y which is visually indistinguishable from x but is misclassified by the network. Typically, the image y is constructed as $y = x + r$, where r is an *adversarial perturbation* that is supposed to be small in a suitable sense (normally, with respect to an ℓ^p norm). Several algorithms have been developed to construct adversarial perturbations, see [9, 18, 13, 16, 6].

Even though such pathological cases are very unlikely to occur in practice, their existence is relevant since malicious attackers may exploit this drawback to fool classifiers or other automatic systems. Further, adversarial perturbations may be constructed in a black-box setting (i.e., without knowing the architecture of the DNN but only its outputs) [19, 17] and also in the physical world [13, 3, 4, 23]. This has motivated the investigation of defenses, i.e., how to make the network invulnerable to such attacks, see [14, 5, 16, 26, 12, 20, 2, 11]. In most cases, adversarial examples are artificially created and then used to retrain the network, which becomes more stable under these types of perturbations.

Most of the work on the construction of adversarial examples and on the design of defense strategies has been conducted in the context of small perturbations r measured in the ℓ^∞ norm. However, this is not necessarily a good measure of image similarity: e.g., for two translated images x and y, the norm of $x - y$ is not small in general, even though x and y will look indistinguishable if the translation is small. Several papers have investigated the construction of adversarial perturbations not designed for norm proximity [21, 23, 4, 7, 27].

In this work, we build up on these ideas and investigate the construction of *adversarial deformations*. In other words, the misclassified image y is not constructed as an additive perturbation $y = x + r$, but as a deformation $y = x \circ (\mathrm{id} + \tau)$, where τ is a vector field defining the transformation. In this case, the similarity is not measured through a norm of $y - x$, but instead through a norm of τ, which quantifies the deformation between y and x.

We develop an efficient algorithm for the construction of adversarial deformations, which we call ADef. It is based on the main ideas of DeepFool [18], and iteratively constructs the smallest deformation to misclassify the image. We test the procedure on MNIST [15] (with a convolutional neural network) and on ImageNet [22] (with Inception-v3 [24] and ResNet-101 [10]). The results show that ADef can succesfully fool the classifiers in the vast majority of cases (around 99%) by using very small and imperceptible deformations.

The results of this work have initially appeared in the master's thesis [8], to which we refer for additional details on the mathematical aspects of this construction (see also [1]). While writing this paper, we have come across [27], in which a similar problem is considered and solved with a different algorithm.

2 Adversarial perturbations

Let $\mathcal{K}$ be a classifier of images consisting of P pixels into $L \geq 2$ categories, i.e. a function from the space of images $X = \mathbb{R}^{cP}$, where $c = 1$ (for grayscale images) or $c = 3$ (for color images), and into the set of labels $\mathcal{L} = \{1, \dots, L\}$. Suppose $x \in X$ is an image that is correctly classified by $\mathcal{K}$ and suppose $y \in X$ is another image that is *imperceptible* from x and such that $\mathcal{K}(y) \neq \mathcal{K}(x)$, then y is said to be an *adversarial example*. The meaning of imperceptibility varies, but generally, proximity in ℓ^p-norm (with $1 \leq p \leq \infty$) is considered to be a sufficient substitute. Thus, an *adversarial perturbation* for an image $x \in X$ is a vector $r \in X$ such that $\mathcal{K}(x + r) \neq \mathcal{K}(x)$ and $\|r\|_p$ is small, where

$$\|r\|_p = \left(\sum_{j=1}^{cP} |r_j|^p \right)^{1/p} \quad \text{if } 1 \leq p < \infty, \text{ and} \quad \|r\|_\infty = \max_{j=1,\dots,cP} |r_j|. \tag{1}$$

Given such a classifier $\mathcal{K}$ and an image x, an adversary may attempt to find an adversarial example y by minimizing $\|x - y\|_p$ subject to $\mathcal{K}(y) \neq \mathcal{K}(x)$, or even subject to $\mathcal{K}(y) = k$ for some *target label* $k \neq \mathcal{K}(x)$.

Different methods for finding minimal adversarial perturbations have been proposed, most notably FGSM [9] and PGD [16] for ℓ^∞, and the DeepFool algorithm [18] for general ℓ^p-norms. We shall now briefly describe the DeepFool algorithm for $p = 2$, as it provides useful insight for the remainder of the present work.

Let $F = (F_1, \dots, F_L) : X \to \mathbb{R}^L$ be the underlying model for the classifier $\mathcal{K}$, such that

$$\mathcal{K}(x) = \operatorname*{arg\,max}_{k=1,\dots,L} F_k(x).$$

Let $x \in X$ be the image of interest with assigned label $l = \mathcal{K}(x)$. Let $k \neq l$ in $\mathcal{L}$ be a target label. For the moment, our goal will be to find an image $y \in X$ such that $F_k(y) \geq F_l(y)$ and with minimal $\|x - y\|_2$.

Define the function $f = F_k - F_l : X \to \mathbb{R}$. Then $f(x) < 0$ and our goal is accomplished if we find an image y, close to x, with $f(y) \geq 0$. Assuming that f is differentiable at x, we can consider the linear approximation

$$f(x + r) \approx f(x) + r \cdot \nabla f(x)$$

for small enough $r \in X$. If $\nabla f(x)$ is non-zero, then with

$$r = -\frac{f(x)}{\|\nabla f(x)\|_2^2} \nabla f(x),$$

we get

$$f(x) + r \cdot \nabla f(x) = f(x) - \frac{f(x)}{\|\nabla f(x)\|_2^2} \nabla f(x) \cdot \nabla f(x) = 0.$$

Thus, if $\|r\|_2 = -f(x) / \|\nabla f(x)\|_2$ is small enough, then

$$f(x + r) \approx 0,$$

and the classifier $\mathcal{K}$ has approximately equal confidence for the image $y = x + r$ to have either label l or k, bringing us near to our goal.

The DeepFool algorithm works by iterating the above approximation, setting $x^{(0)} = x$ and $x^{(n)} = x^{(n-1)} + r^{(n)}$ with

$$r^{(n)} = -\frac{F_k(x^{(n)}) - F_l(x^{(n)})}{\left\|\nabla F_k(x^{(n)}) - \nabla F_l(x^{(n)})\right\|_2^2} \left(\nabla F_k(x^{(n)}) - \nabla F_l(x^{(n)})\right)$$

and allowing the target label k to change in each iteration to minimize the norm of $r^{(n)}$. The process is terminated when $\mathcal{K}(x^{(n)}) \neq l$ and the algorithm outputs an adversarial example $y = x^{(n)} = x + r^{(1)} + \ldots + r^{(n)}$. The process may also converge to a decision boundary of the classifier, in which case the total perturbation $r^* = r^{(1)} + \ldots + r^{(n)}$ can be multiplied by a factor $1 + \eta$ where η is some small positive real number, to ensure that $y = x + (1 + \eta) r^*$ lies on the opposite side of the boundary from x.

Digital images have pixel values in a bounded range such as from 0 to 255. Perturbing an image with an algorithm such as DeepFool does not guarantee that the resulting vector is a valid image with pixel values constrained to that range. Hence, an adversarial example from DeepFool may well lie outside the region in X on which the classifier has been trained and can realistically receive inputs. In [1], we discuss the impact on the success rate of the DeepFool algorithm when the perturbed image is projected on the set of valid images.

3 Adversarial deformations

We now make clear what is meant by a deformation of an image. It simplifies the discussion to model images as functions $\xi : [0,1]^2 \to \mathbb{R}^c$ (with $c = 1$ or $c = 3$) instead of discrete vectors x in $\mathbb{R}^{cP}$. In this setting, perturbing an image $\xi : [0,1]^2 \to \mathbb{R}^c$ corresponds to adding to it another function $\rho : [0,1]^2 \to \mathbb{R}^c$ with a small L^p-norm, defined analogously to the discrete case (1) by

$$\|\rho\|_{L^p} = \left(\int_{[0,1]^2} \|\rho(u)\|_p^p \, \mathrm{d}u \right)^{1/p} \quad \text{if } 1 \leq p < \infty, \text{ and} \quad \|\rho\|_{L^\infty} = \operatorname*{ess\,sup}_{u \in [0,1]^2} \|\rho(u)\|_\infty . \tag{2}$$

While any transformation of an image ξ can be written as a perturbation $\xi + \rho$, we shall restrict ourselves to a particular class of transformations. A *deformation* with respect to a vector field $\sigma : [0,1]^2 \to \mathbb{R}^2$ is a transformation of the form $\xi \mapsto \xi^\sigma$, where for any image $\xi : [0,1]^2 \to \mathbb{R}^c$, the image $\xi^\sigma : [0,1]^2 \to \mathbb{R}^c$ is defined by

$$\xi^\sigma(u) = \xi\left(u + \sigma(u)\right) \quad \text{for all } u \in [0,1]^2,$$

extending ξ by zero outside of $[0,1]^2$. Deformations capture many natural image transformations. For example, a translation of the image ξ by a vector $v \in \mathbb{R}^2$ is a deformation with respect to the constant vector field $\sigma = v$. If v is small, the images ξ and ξ^v may look similar, but the corresponding perturbation $\rho = \xi^v - \xi$ may be arbitrarily large in the aforementioned L^p-norms. Figure 1 shows three minor deformations, all of which yield large L^∞-norms.

We will now describe our procedure for finding deformations that will lead a classifier to yield an output different from the original label. Our derivations are based on the setting that we have briefly sketched above for a continuous model, but for the sake of simplicity, we wish to only describe it in a discrete setting in what follows. For the formal derivation we refer to [1].

We consider square images of $W \times W$ pixels and define the space of images to be $X = \mathbb{R}^{c \times W \times W}$. A discrete vector field is a function $\tau : \{1, \ldots, W\}^2 \to \mathbb{R}^2$. In what follows we will only consider

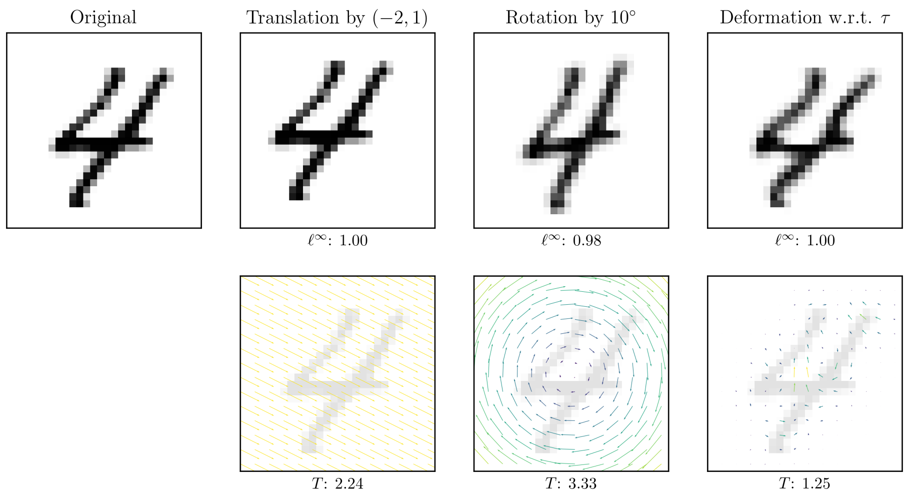

Figure 1: **First row:** The original 28×28 pixel image from the MNIST database, and the same image translated by $(-2, 1)$, rotated by an angle of $10°$, and deformed w.r.t. an arbitrary smooth vector field τ. The ℓ^∞-norm of the corresponding perturbation is shown under each deformed image. The pixel values range from 0 (white) to 1 (black), so the deformed images all lie far from the original image in the l^∞-norm. **Second row:** The vector fields corresponding to the above deformations and their T-norms.

the set T of vector fields that do not move points on the grid $\{1,\dots,W\}^2$ outside of $[1,W]^2$. More precisely,

$$T := \left\{\tau : \{1,\dots,W\}^2 \to \mathbb{R}^2 \mid \tau(s,t) + (s,t) \in [1,W]^2 \text{ for all } s,t \in \{1,\dots,W\}\right\}.$$

An image $x = (x_1,\dots,x_c) \in X$ can be viewed as the collection of values of a function $\xi = (\xi_1,\dots,\xi_c) : [0,1]^2 \to \mathbb{R}^c$ on a regular grid $\{1/(W+1),\dots,W/(W+1)\}^2 \subseteq [0,1]^2$, i.e. $x_{i,s,t} = \xi_i(s/(W+1), t/(W+1))$ for $i = 1,\dots,c$ and $s,t = 1,\dots,W$. Such a function ξ can be computed by interpolating from x. Likewise, a discrete vector field $\tau \in T$ can be viewed as an evaluation of a vector field mapping from $[0,1]^2$ to $\mathbb{R}^2$ on the same grid. Thus, the deformation of an image x with respect to the discrete vector field τ can be defined as the discrete deformed image x^τ in X by

$$x^\tau_{i,s,t} = \xi_i\left(\frac{(s,t)+\tau(s,t)}{W+1}\right), \tag{3}$$

for $i = 1,\dots,c$ and $s,t = 1,\dots,W$, or $x^\tau_i = \xi_i \circ ((\mathrm{id}+\tau)/(W+1))$ in short.

It is not straightforward to measure the size of a deformation such that it captures the visual difference between the original image x and its deformed counterpart x^τ. We will use the size of the corresponding vector field, τ, in the norm defined by

$$\|\tau\|_T = \max_{s,t=1,\dots,W} \|\tau(s,t)\|_2 \tag{4}$$

as a proxy. The ℓ^p-norms defined in (1), adapted to vector fields, can be used as well. (We remark, however, that none of these norms define a distance between x and x^τ, since two vector fields $\tau,\sigma \in T$ with $\|\tau\|_T \neq \|\sigma\|_T$ may produce the same deformed image $x^\tau = x^\sigma$.)

As in section 2, we let $\mathcal{K} : X \to \mathcal{L}$ be a classifier and let $F = (F_1,\dots,F_L) : X \to \mathbb{R}^L$ be the underlying model, such that

$$\mathcal{K}(x) = \underset{k=1,\dots,L}{\arg\max}\, F_k(x).$$

Let $x \in X$ be an image and fix $\xi : [0,1]^2 \to \mathbb{R}^c$ obtained by interpolation from x, such that $x_{i,s,t} = \xi_i(s/(W+1), t/(W+1))$ for $i = 1,\dots,c$ and $s,t = 1,\dots,W$. Let $l = \mathcal{K}(x)$ denote the true label of x, let $k \in \mathcal{L}$ be a target label and set $f = F_k - F_l$. We assume that x does not lie on a decision boundary, so that we have $f(x) < 0$.

We define the function $g : T \to \mathbb{R}, \tau \mapsto f(x^\tau)$ and note that $g(0) = f(x^0) = f(x) < 0$. Our goal is to find a small vector field $\tau \in T$ such that $g(\tau) = f(x^\tau) \geq 0$. Under suitable assumptions on ξ and f, the derivative of g at 0, $\mathrm{D}_0 g : X \to \mathbb{R}$, is expressible in terms of the known quantities $\nabla f(x)$ and $\nabla\xi_1, \ldots, \nabla\xi_c$. Here, $\xi_1, \ldots, \xi_c$ are the color components of ξ and their derivatives can be approximated by finite differences. Similarly to the derivation in section 2, we can use a linear approximation of g around the zero vector field as a guide:

$$g(\tau) \approx g(0) + (\mathrm{D}_0 g)\,\tau \tag{5}$$

for small enough $\tau \in T$. Hence, if τ is a vector field such that

$$(\mathrm{D}_0 g)\,\tau = -g(0) \tag{6}$$

and $\|\tau\|_T$ is small, then the classifier $\mathcal{K}$ has approximately equal confidence for the deformed image x^τ to have either label l or k.

This is a scalar equation with unknown in T, and so has infinitely many solutions. In order to select τ with small norm, we solve it in the least-squares sense. As derived in [1], we have that $(\mathrm{D}_0 g)\,\tau = \sum_{s,t=1}^{W} \alpha(s,t) \cdot \tau(s,t)$, where the discrete vector field $\alpha : \{1, \ldots, W\}^2 \to \mathbb{R}^2$ is given by

$$\alpha(s,t) = \frac{1}{W+1} \sum_{i=1}^{c} \left(\nabla f(x)\right)_{i,s,t} \cdot \nabla\xi_i \left(\frac{s}{W+1}, \frac{t}{W+1}\right). \tag{7}$$

Let

$$A = \sum_{s,t=1}^{W} \|\alpha(s,t)\|_2^2 = \|\alpha\|_{\ell^2}^2$$

be the sum of the squares of the entries of $\alpha = (\alpha_1, \alpha_2)$. Then, if $A \neq 0$, the discrete vector field $\tau : \{1, \ldots, W\}^2 \to \mathbb{R}^2$, chosen as

$$\tau = -\frac{f(x)}{A}\,\alpha, \tag{8}$$

corresponds to a solution of equation (6). Finally, we define the deformed image $x^\tau \in X$ according to (3).

One might like to impose some degree of smoothness on the deforming vector field. In fact, it suffices to search in the range of a smoothing operator $\mathcal{S} : T \to T$. However, this essentially amounts to applying $\mathcal{S}$ to the solution from the larger search space T. Let $\tilde{\alpha} = \mathcal{S}\alpha = (\varphi * \alpha_1, \varphi * \alpha_2)$, where $\mathcal{S}$ denotes the componentwise application of a two-dimensional Gaussian filter φ (of any standard deviation), and let $\tilde{A} = \|\tilde{\alpha}\|_{\ell^2}^2$. Then the vector field

$$\tilde{\tau} = -\frac{f(x)}{\tilde{A}}\,\mathcal{S}\tilde{\alpha} = -\frac{f(x)}{\tilde{A}}\,\mathcal{S}^2\alpha$$

also satisfies (6), since $\mathcal{S}$ is self-adjoint. We can hence replace τ by $\tilde{\tau}$ to obtain a smooth deformation of the image x.

We proceed as in section 2, and iterate the deformation process until the deformed image is misclassified. More explicitly, let $x^{(0)} = x$ and for $n \geq 1$ let $\tau^{(n)}$ solve (8) for $x^{(n-1)}$. Then we can define the iteration as $x^{(n)} = x^{(n-1)} \circ (\mathrm{id} + \tau^{(n)})$. The algorithm terminates and outputs an adversarial example $y = x^{(n)}$ if $\mathcal{K}(x^{(n)}) \neq l$. The iteration also terminates if $x^{(n)}$ lies on a decision boundary of $\mathcal{K}$, in which case we propose to introduce an overshoot factor $1 + \eta$ on the *total* deforming vector field. Provided that the number of iterations is moderate, the total vector field can be well approximated by $\tau^* = \tau^{(1)} + \cdots + \tau^{(n)}$ and the process can be altered to output the deformed image $y = x \circ (\mathrm{id} + (1+\eta)\tau^*)$ instead.

The target label k may be chosen in each iteration to minimize the vector field to obtain a better approximation in the linearization (5). More precisely, for a candidate set of labels $k_1, \ldots, k_m$, we compute the corresponding vectors fields $\tau_1, \ldots, \tau_m$ and select

$$k = \underset{j=1,\ldots,m}{\arg\min}\, \|\tau_j\|_T.$$

The candidate set consists of the labels corresponding to the indices of the m smallest entries of $F - F_l$, in absolute value.

```
Algorithm ADef
Input: Classification model $F$, image $x$, correct label $l$, candidate labels $k_1, \dots, k_m$
Output: Deformed image $y$
  Initialize $y \leftarrow x$
  while $\mathcal{K}(y) = l$ do
      for $j = 1, \dots, m$ do
          $\alpha_j \leftarrow \sum_{i=1}^{c} \big((\nabla F_{k_j})_i - (\nabla F_l)_i\big) \cdot \nabla y_i$
          $\tau_j \leftarrow -\frac{F_{k_j}(y) - F_l(y)}{\|\mathcal{S}\alpha\|_{\ell^2}^2} \mathcal{S}^2 \alpha_j$
      end for
      $i \leftarrow \arg\min_{j=1,\dots,m} \|\tau_j\|_T$
      $y \leftarrow y \circ (\mathrm{id} + \tau_i)$
  end while
  return $y$
```

By equation (7), given that ∇f is moderate, the deforming vector field takes small values wherever ξ has a small derivative. This means that the vector field will be concentrated on the edges in the image x (see e.g. the first row of figure 3). Further, note that the result of a deformation is always a valid image in the sense that it does not violate the pixel value bounds. This is not guaranteed for the perturbations computed with DeepFool (cf. Section 2).

4 Experiments

We evaluate the performance of ADef by applying the algorithm to classifiers trained on the MNIST [15] and ImageNet [22] datasets. Below, we briefly describe the setup of the experiments and in table 1 we summarize their results.

MNIST: We train a four-layer convolutional neural network. The first layer consists of 32 convolutional filters of size 5×5, which are followed by 2×2 max-pooling and a rectifier activation function. The second layer has 64 filters and is otherwise identical to the first. The third layer is a fully connected layer into dimension 1024 with a rectifier activation function, and the final layer is linear with output dimension 10. We shall call this model MNIST-CNN. We use ADef to produce adversarial deformations of the images in the test set. The algorithm is configured to pursue any label different from the correct label (all incorrect labels are candidate labels). It performs smoothing by a Gaussian filter of standard deviation $1/2$, and it overshoots by $\eta = 2/10$ whenever it converges to a decision boundary.

ImageNet: We apply ADef to pretrained Inception-v3 [24] and ResNet-101 [10] models to generate adversarial deformations for the images in the ILSVRC2012 validation set. The images are preprocessed by first scaling so that the smaller axis has 299 pixels for the Inception model and 224 pixels for ResNet, and then they are center-cropped to a square image. The algorithm is set to focus only on the label of second highest probability. It employs a Gaussian filter of standard deviation 1 and an overshoot factor $\eta = 1/10$.

We only consider inputs that are correctly classified by the model in question, and, since $\tau^* = \tau^{(1)} + \cdots + \tau^{(n)}$ approximates the total deforming vector field, we declare ADef to be successful if its output is misclassified and $\|\tau^*\|_T \leq \varepsilon$, where we choose $\varepsilon = 3$. Observe that, by (4), a deformation with respect to a vector field τ does not displace any pixel further away from its original position than $\|\tau\|_T$. Hence, for high resolution images, the choice $\varepsilon = 3$ indeed produces small deformations if the vector fields are smooth. From figure 2, we can see how the success rate of ADef depends on the choice of ε.

When searching for an adversarial example, one usually searches for a perturbation with ℓ^∞-norm smaller than some small number $\varepsilon > 0$. Common choices of ε range from $1/10$ to $3/10$ for MNIST classifiers [9, 16, 12, 26, 11] and $2/255$ to $16/255$ for ImageNet classifiers [9, 14, 26, 11]. Table 1 shows that on average, the perturbations obtained by ADef are quite large compared to those constraints. However, as can be seen in figure 3, the relatively high resolution images of the ImageNet dataset can be deformed into adversarial examples that, while corresponding to large perturbations, are not visibly different to the original images.

Model	Accuracy	ADef success	Avg. $\|\tau^*\|_T$	Avg $\|r\|_\infty$	Avg. # iterations
MNIST-CNN	99.41%	99.90%	1.0864	0.6927	9.779
Inception-v3	77.56%	98.94%	0.5984	0.2039	4.050
ResNet-101	76.97%	99.78%	0.5561	0.1882	4.176

Table 1: The results of applying ADef to the images in the MNIST test set (first row) and the ILSVRC2012 validation set (second and third rows). The accuracy of the Inception and ResNet models is defined as the top-1 accuracy on the center-cropped and resized images. The success rate of ADef is shown as a percentage of the correctly classified inputs. The pixel range is scaled to $[0, 1]$, so the perturbation $r = y - x$, where x is the input and y the output of ADef, has values in $[-1, 1]$. The averages in the three last columns are computed over the set of images on which ADef is successful.

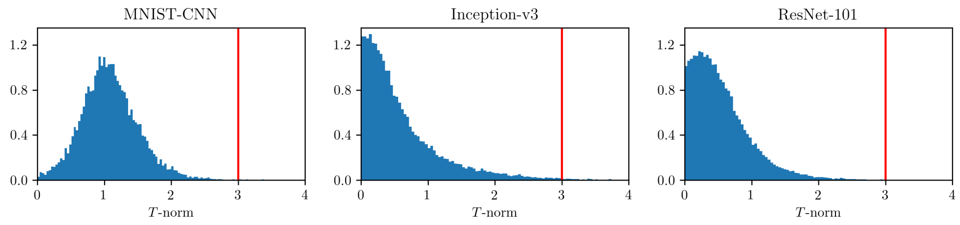

Figure 2: The (normalized) distribution of $\|\tau^*\|_T$ from the experiment. Deformations that fall to the left of the vertical line at $\varepsilon = 3$ are considered successful.

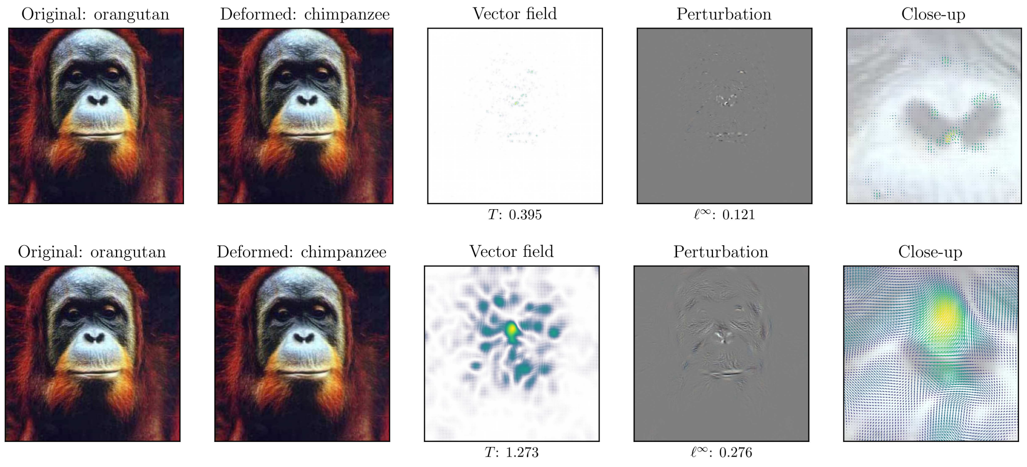

Figure 3: Sample deformations for the Inception-v3 model. The vector fields and perturbations have been amplified for visualization. **First row:** An image from the ILSVRC2012 validation set, the output of ADef with a Gaussian filter of standard deviation 1, the corresponding vector field and perturbation. The rightmost image is a close-up of the vector field around the nose of the ape. **Second row:** A larger deformation of the same image, obtained by using a wider Gaussian filter (standard deviation 6) for smoothing.

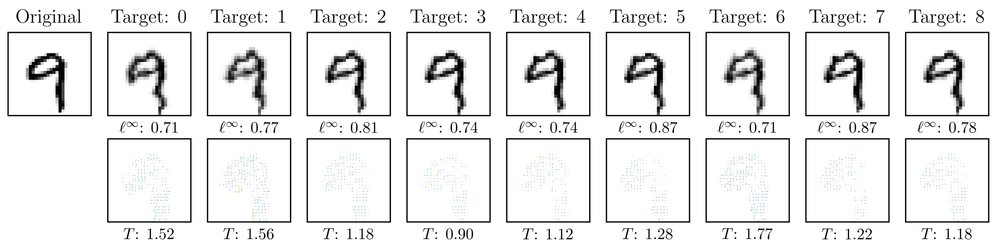

Figure 4: **First row:** The original image and deformed images produced by restricting ADef to the target labels 0 to 8. The ℓ^∞-norms of the corresponding perturbations are shown under the deformed images. **Second row:** The vector fields corresponding to the deformations and their T-norms.

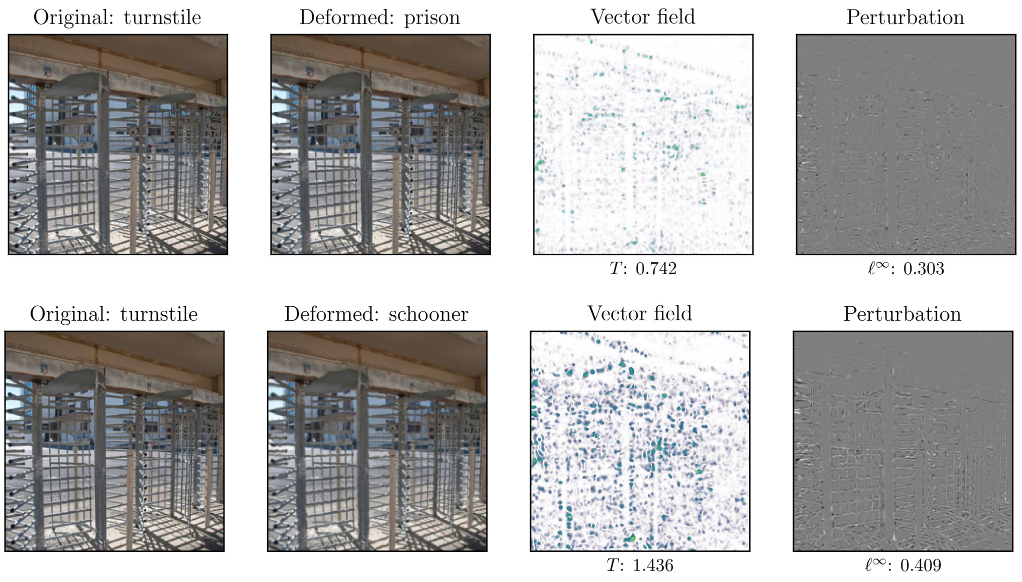

Figure 5: Untargeted vs. targeted attack on the ResNet-101 model. An image from the ILSVRC2012 validation set deformed to the labels of second highest (first row) and lowest (second row) probabilities (out of 1,000) for the original image. The vector fields and perturbations have been amplified for visualization.

ADef can also be used for *targeted* adversarial attacks, by restricting the deformed image to have a particular target label instead of any label which yields the optimal deformation. Figure 4 demonstrates the effect of choosing different target labels for a given MNIST image, and figure 5 shows the result of targeting the label of lowest probability for an image from the ImageNet dataset.

5 Conclusion

In this work, we proposed a new efficient algorithm, ADef, to construct a new type of adversarial attacks for DNN image classifiers. The procedure is iterative and in each iteration takes a gradient descent step to deform the previous iterate in order to push to a decision boundary.

We demonstrated that with almost imperceptible deformations, state-of-the art classifiers can be fooled to misclassify with a high success rate of ADef. This suggests that networks are vulnerable to different types of attacks and that simply training the network on a specific class of adversarial examples might not form a sufficient defense strategy.

Acknowledgments. The authors would like to thank Helmut Bölcskei and Thomas Wiatowski for fruitful discussions.

References

[1] R. Alaifari, G. S. Alberti, and T. Gauksson. Adversarial deformations for deep neural networks. *in preparation*, 2018.

[2] A. Athalye, N. Carlini, and D. Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. *arXiv preprint arXiv:1802.00420*, 2018.

[3] A. Athalye and I. Sutskever. Synthesizing robust adversarial examples. *arXiv preprint arXiv:1707.07397*, 2017.

[4] T. B. Brown, D. Mané, A. Roy, M. Abadi, and J. Gilmer. Adversarial patch. *arXiv preprint arXiv:1712.09665*, 2017.

[5] N. Carlini and D. Wagner. Adversarial examples are not easily detected: Bypassing ten detection methods. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pages 3–14. ACM, 2017.

[6] N. Carlini and D. Wagner. Towards evaluating the robustness of neural networks. In *IEEE Symposium on Security and Privacy (SP)*, pages 39–57. IEEE, 2017.

[7] L. Engstrom, D. Tsipras, L. Schmidt, and A. Madry. A Rotation and a Translation Suffice: Fooling CNNs with Simple Transformations. *arXiv preprint arXiv:1712.02779*, 2017.

[8] T. Gauksson. Adversarial perturbations and deformations for convolutional neural networks. `https://www.research-collection.ethz.ch/handle/20.500.11850/258550/`, 2017.

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

[10] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[11] H. Kannan, A. Kurakin, and I. Goodfellow. Adversarial logit pairing. *arXiv preprint arXiv:1803.06373*, 2018.

[12] J. Z. Kolter and E. Wong. Provable defenses against adversarial examples via the convex outer adversarial polytope. *arXiv preprint arXiv:1711.00851*, 2017.

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

[14] A. Kurakin, I. Goodfellow, and S. Bengio. Adversarial machine learning at scale. *arXiv preprint arXiv:1611.01236*, 2016.

[15] Y. LeCun. The MNIST database of handwritten digits. `http://yann.lecun.com/exdb/mnist/`.

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

[17] S. M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard. Universal adversarial perturbations. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 86–94, July 2017.

[18] S. M. Moosavi Dezfooli, A. Fawzi, and P. Frossard. DeepFool: a simple and accurate method to fool deep neural networks. In *Proceedings of 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, number EPFL-CONF-218057, 2016.

[19] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, pages 506–519. ACM, 2017.

[20] A. Raghunathan, J. Steinhardt, and P. Liang. Certified defenses against adversarial examples. *arXiv preprint arXiv:1801.09344*, 2018.

[21] A. Rozsa, E. M. Rudd, and T. E. Boult. Adversarial diversity and hard positive generation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 25–32, 2016.

[22] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[23] M. Sharif, S. Bhagavatula, L. Bauer, and M. K. Reiter. Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 1528–1540. ACM, 2016.

[24] 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 (CVPR)*, pages 2818–2826, 2016.

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

[26] F. Tramèr, A. Kurakin, N. Papernot, D. Boneh, and P. McDaniel. Ensemble adversarial training: Attacks and defenses. *arXiv preprint arXiv:1705.07204*, 2017.

[27] C. Xiao, J.-Y. Zhu, B. Li, W. He, M. Liu, and D. Song. Spatially transformed adversarial examples. *arXiv preprint arXiv:1801.02612*, 2018.