ADEF: AN ITERATIVE ALGORITHM TO CONSTRUCT ADVERSARIAL DEFORMATIONS

Anonymous authors
Paper under double-blind review

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 convolutional neural networks and on ImageNet with Inception-v3 and ResNet-101.

1 INTRODUCTION

In a first observation in Szegedy et al. (2013) it was found that deep neural networks exhibit unstable behavior to small perturbations in the input. For the task of image classification this means that 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 then, 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 Goodfellow et al. (2014); Moosavi Dezfooli et al. (2016); Kurakin et al. (2017b); Madry et al. (2018); Carlini & Wagner (2017b) and the review paper Akhtar & Mian (2018).

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) (Papernot et al., 2017; Moosavi-Dezfooli et al., 2017) and also in the physical world (Kurakin et al., 2017b; Athalye & Sutskever, 2017; Brown et al., 2017; Sharif et al., 2016). This has motivated the investigation of defenses, i.e., how to make the network invulnerable to such attacks, see Kurakin et al. (2017a); Carlini & Wagner (2017a); Madry et al. (2018); Tramèr et al. (2018); Wong & Kolter (2018); Raghunathan et al. (2018); Athalye et al. (2018); Kannan et al. (2018). 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 (Rozsa et al., 2016; Sharif et al., 2016; Brown et al., 2017; Engstrom et al., 2017; Xiao et al., 2018).

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 (Moosavi Dezfooli et al., 2016), and iteratively constructs the smallest deformation to misclassify the image. We test the procedure on MNIST (LeCun) (with convolutional neural networks) and on ImageNet (Russakovsky et al., 2015) (with Inception-v3 (Szegedy et al., 2016) and ResNet-101 (He et al., 2016)). 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. We also test our adversarial attacks on adversarially trained networks for MNIST.

The results of this work have initially appeared in the master's thesis Anonymous (2017), to which we refer for additional details on the mathematical aspects of this construction[1]. While writing this paper, we have come across Xiao et al. (2018), in which a similar problem is considered and solved with a different algorithm. Whereas in Xiao et al. (2018) the authors use a second order solver to find a deforming vector field, we show how a first order method can be formulated efficiently and justify a smoothing operation, independent of the optimization step. We report, for the first time, success rates for adversarial attacks with deformations on ImageNet. The topic of deformations has also come up in Jaderberg et al. (2015), in which the authors introduce a class of learnable modules that deform inputs in order to increase the performance of existing DNNs.

2 Adversarial deformations

2.1 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 (Goodfellow et al., 2014) and PGD (Madry et al., 2018) for ℓ^∞, and the DeepFool algorithm (Moosavi Dezfooli et al., 2016) for general ℓ^p-norms.

2.2 Deformations

Instead of constructing adversarial perturbations, we intend to fool the classifier by small deformations of correctly classified images. Our procedure is in the spirit of the DeepFool algorithm. Before we explain it, let us first clarify what we mean by a deformation of an image. The discussion is at first more intuitive if we 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.

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 $\tau : [0,1]^2 \to \mathbb{R}^2$ is a transformation of the form $\xi \mapsto \xi^\tau$, where for any image $\xi : [0,1]^2 \to \mathbb{R}^c$, the image $\xi^\tau : [0,1]^2 \to \mathbb{R}^c$ is defined by

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

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

[1] Appendix C contains an anonymized version of the relevant part of Anonymous (2017).

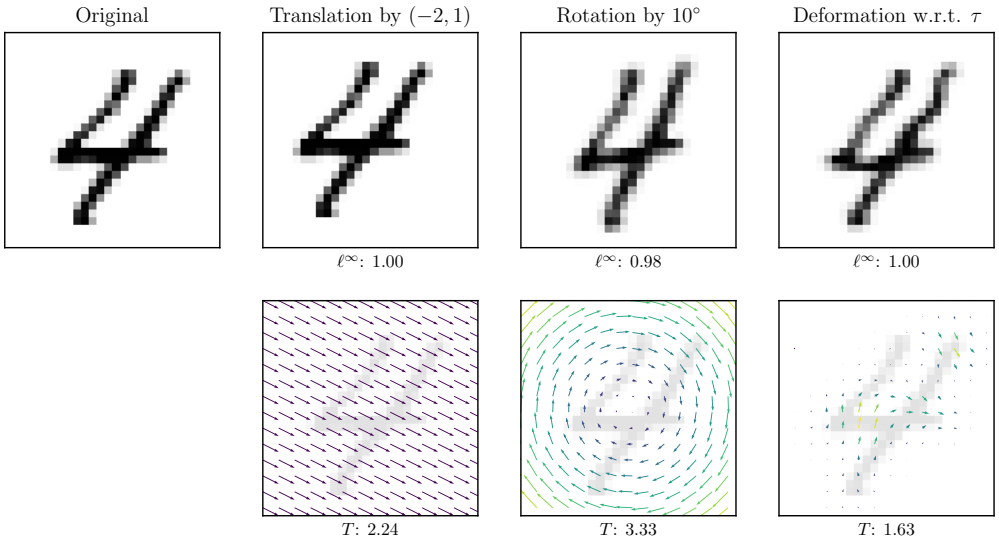

Figure 1: **First row:** The original 28×28 pixel image from the MNIST database, and the same image translated by $(-2, 1)$, rotated by an angle of $10°$, and deformed w.r.t. an arbitrary smooth vector field τ. The ℓ^∞-norm of the corresponding perturbation is shown under each deformed image. The pixel 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 (cf. equation (3)).

constant vector field $\tau = v$. 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.

In the discrete setting, deformations are implemented as follows. We consider square images of $W \times W$ pixels and define the space of images to be $X = \left(\mathbb{R}^{W \times W}\right)^c$. A discrete vector field is a function $\tau : \{1, \ldots, W\}^2 \to \mathbb{R}^2$. In what follows we will only consider the set T of vector fields that do not move points on the grid $\{1, \ldots, W\}^2$ outside of $[1, W]^2$. More precisely,

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

An image $x \in X$ can be viewed as the collection of values of a function $\xi : [0,1]^2 \to \mathbb{R}^c$ on a regular grid $\{{}^1/_{(W+1)}, \ldots, {}^W/_{(W+1)}\}^2 \subseteq [0,1]^2$, i.e. $x_{s,t} = \xi({}^s/_{(W+1)}, {}^t/_{(W+1)})$ for $s, t = 1, \ldots, W$. Such a function ξ can be computed by interpolating from x. 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

$$x^\tau_{s,t} = \xi\left(\frac{(s,t) + \tau(s,t)}{W+1}\right), \qquad s, t \in \{1, \ldots, W\}. \tag{2}$$

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

$$\|\tau\|_T = \max_{s,t=1,\ldots,W} \|\tau(s,t)\|_2 \tag{3}$$

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 $x^\tau = x^\sigma$.)

2.3 THE ALGORITHM ADEF

We will now describe our procedure for finding deformations that will lead a classifier to yield an output different from the original label.

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

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

Let $x \in X$ be the image of interest and fix $\xi : [0,1]^2 \to \mathbb{R}^c$ obtained by interpolation from x. 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 $g(\tau) = f(x^\tau) \geq 0$. We can use a linear approximation of g around the zero vector field as a guide:

$$g(\tau) \approx g(0) + (\mathrm{D}_0 g)\, \tau \tag{4}$$

for small enough $\tau \in T$ and $D_0 g : T \to \mathbb{R}$ the derivative of g at $\tau = 0$. Hence, if τ is a vector field such that

$$(\mathrm{D}_0 g)\, \tau = -g(0) \tag{5}$$

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.

In view of (2), we have $\frac{\partial x^\tau}{\partial \tau}|_{\tau=0}(s,t) = \frac{1}{W+1} \nabla \xi \left(\frac{(s,t)}{W+1} \right) \in \mathbb{R}^{c \times 2}$. Thus, by applying the chain rule to $g(\tau) = f(x^\tau)$, we obtain that its derivative at $\tau = 0$ can, with a slight abuse of notation, be identified with the vector field

$$\mathrm{D}_0 g(s,t) = \frac{1}{W+1} \big(\nabla f(x)\big)_{s,t} \nabla \xi \left(\frac{(s,t)}{W+1} \right), \tag{6}$$

where $\big(\nabla f(x)\big)_{s,t} \in \mathbb{R}^{1 \times c}$ is the derivative of f in x calculated at (s,t). With this, $(\mathrm{D}_0 g)\, \tau$ stands for $\sum_{s,t=1}^{W} \mathrm{D}_0 g(s,t) \cdot \tau(s,t)$, and the solution to (5) in the least-square sense is given by

$$\tau = -\frac{f(x)}{\sum_{s,t=1}^{W} |\mathrm{D}_0 g(s,t)|^2}\, \mathrm{D}_0 g. \tag{7}$$

Finally, we define the deformed image $x^\tau \in X$ according to (2).

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 $\alpha = \mathcal{S}(\mathrm{D}_0 g) = \varphi * (\mathrm{D}_0 g)$, where $\mathcal{S}$ denotes the componentwise application of a two-dimensional Gaussian filter φ (of any standard deviation). Then the vector field

$$\tilde{\tau} = -\frac{f(x)}{\sum_{s,t=1}^{W} |\alpha(s,t)|^2}\, \mathcal{S}\alpha = -\frac{f(x)}{\sum_{s,t=1}^{W} |\alpha(s,t)|^2}\, \mathcal{S}^2(\mathrm{D}_0 g)$$

also satisfies (5), since $\mathcal{S}$ is self-adjoint. We can hence replace τ by $\tilde{\tau}$ to obtain a smooth deformation of the image x.

We iterate the deformation process until the deformed image is misclassified. More explicitly, let $x^{(0)} = x$ and for $n \geq 1$ let $\tau^{(n)}$ be given by (7) 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 $\tau^* = \tau^{(1)} + \dots + \tau^{(n)}$ 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 (4). 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

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

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 $k_1, \ldots, k_m$
Output: Deformed image y

```
Initialize y ← x
while K(y) = l do
    for j = 1, ..., m do
        α_j ← S ( Σ_{i=1}^{c} ((∇F_{k_j})_i − (∇F_l)_i) · ∇y_i )
        τ_j ← − (F_{k_j}(y) − F_l(y)) / ||α_j||²_{ℓ2} S α_j
    end for
    i ← arg min_{j=1,...,m} ||τ_j||_T
    y ← y ∘ (id + τ_i)
end while
return y
```

By equation (6), 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 2). 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.

3 Experiments

3.1 Setup

We evaluate the performance of ADef by applying the algorithm to classifiers trained on the MNIST (LeCun) and ImageNet (Russakovsky et al., 2015) datasets. Below, we briefly describe the setup of the experiments and in tables 1 and 2 we summarize their results.

MNIST: We train two convolutional neural networks based on architectures that appear in Madry et al. (2018) and Tramèr et al. (2018) respectively. The network MNIST-A consists of two convolutional layers of sizes $32 \times 5 \times 5$ and $64 \times 5 \times 5$, each followed by 2×2 max-pooling and a rectifier activation function, a fully connected layer into dimension 1024 with a rectifier activation function, and a final linear layer with output dimension 10. The network MNIST-B consists of two convolutional layers of sizes $128 \times 3 \times 3$ and $64 \times 3 \times 3$ with a rectifier activation function, a fully connected layer into dimension 128 with a rectifier activation function, and a final linear layer with output dimension 10. During training, the latter convolutional layer and the former fully connected layer of MNIST-B are subject to dropout of drop probabilities $1/4$ and $1/2$. 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 (Szegedy et al., 2016) and ResNet-101 (He et al., 2016) 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 $\eta = 1/10$.

Table 1: The results of applying ADef to the images in the MNIST test set and the ILSVRC2012 validation set. 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. Recall the definition of the vector field norm in equation (3).

Model	Accuracy	ADef success	Avg. $\|\tau^*\|_T$	Avg. $\|r\|_\infty$	Avg. # iterations
MNIST-A	98.99%	99.85%	1.1950	0.7455	7.002
MNIST-B	98.91%	99.51%	1.0841	0.7654	4.422
Inception-v3	77.56%	98.94%	0.5984	0.2039	4.050
ResNet-101	76.97%	99.78%	0.5561	0.1882	4.176

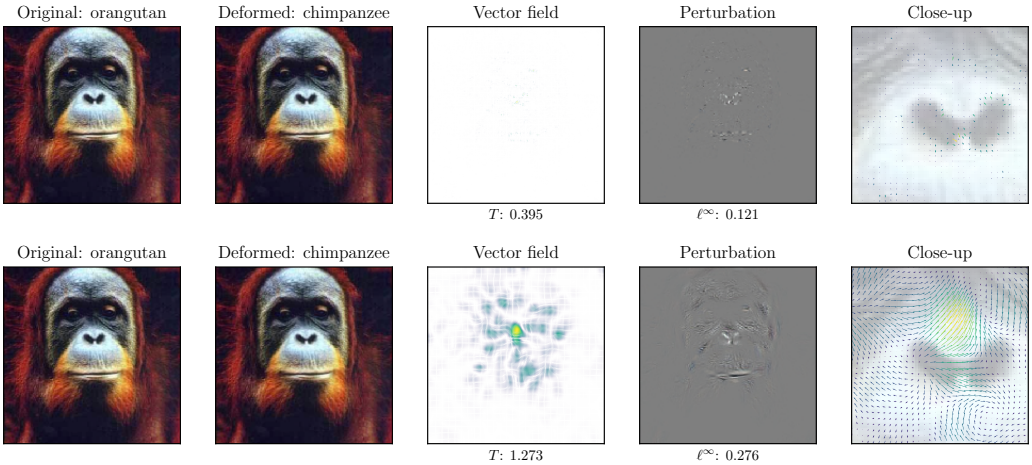

Figure 2: Sample deformations for the Inception-v3 model. The vector fields and perturbations have been amplified for visualization. **First row:** An image from the ILSVRC2012 validation set, the output of ADef with a Gaussian filter of standard deviation 1, the corresponding vector field and perturbation. 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.

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 (3), 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. In appendix A, we illustrate 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 (Goodfellow et al., 2014; Madry et al., 2018; Wong & Kolter, 2018; Tramèr et al., 2018; Kannan et al., 2018) and $2/255$ to $16/255$ for ImageNet classifiers (Goodfellow et al., 2014; Kurakin et al., 2017a; Tramèr et al., 2018; Kannan et al., 2018). 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 2, 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 from the original images. In appendix B, we give more examples of adversarially deformed images.

Table 2: Success rates for PGD and ADef attacks on adversarially trained networks.

Model	Adv. training	Accuracy	PGD success	ADef success
MNIST-A	PGD	98.36%	5.81%	6.67%
	ADef	98.95%	100.00%	54.16%
MNIST-B	PGD	98.74%	5.84%	20.35%
	ADef	98.79%	100.00%	45.07%

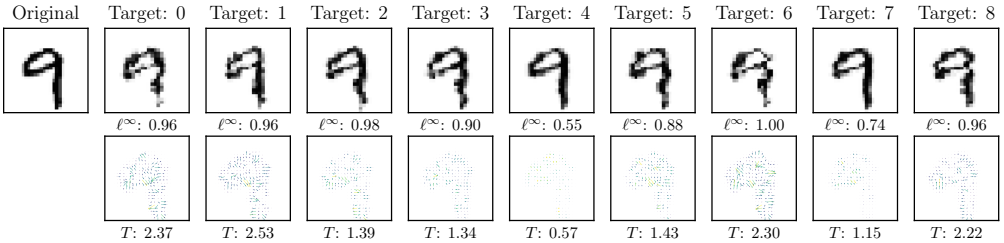

Figure 3: Targeted ADef against MNIST-A. **First row:** The original image and deformed images produced by restricting ADef to the target labels 0 to 8. The ℓ^∞-norms of the corresponding perturbations are shown under the deformed images. **Second row:** The vector fields corresponding to the deformations and their T-norms.

3.2 Adversarial training

In addition to training MNIST-A and MNIST-B on the original MNIST data, we train independent copies of the networks using the adversarial training procedure described by Madry et al. (2018). That is, before each step of the training process, the input images are adversarially perturbed using the PGD algorithm. This manner of training provides increased robustness against adversarial perturbations of low ℓ^∞-norm. Moreover, we train networks using ADef instead of PGD as an adversary. In table 2 we show the results of attacking these adversarially trained networks, using ADef on the one hand, and PGD on the other. We use the same configuration for ADef as above, and for PGD we use 40 iterations, step size $1/100$ and $3/10$ as the maximum ℓ^∞-norm of the perturbation. Interestingly, using these configurations, the networks trained against PGD attacks are more resistant to adversarial deformations than those trained against ADef.

3.3 Targeted attacks

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 3 demonstrates the effect of choosing different target labels for a given MNIST image, and figure 4 shows the result of targeting the label of lowest probability for an image from the ImageNet dataset.

4 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. Given this vulnerability of neural networks to deformations, we wish to study in future work how ADef can help for designing possible defense strategies. Furthermore, we also showed initial results on fooling adversarially trained networks.

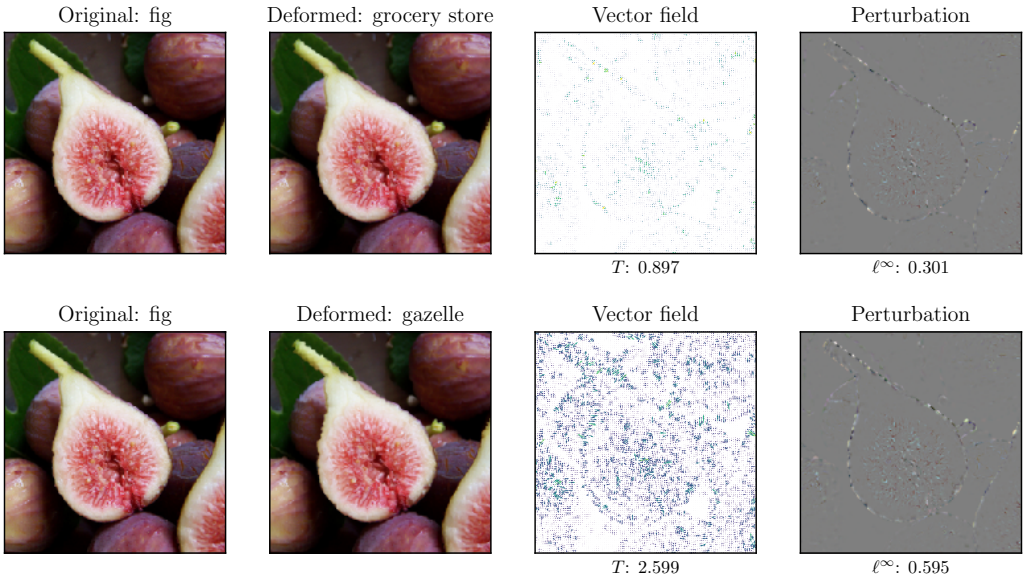

Figure 4: Untargeted vs. targeted attack on the ResNet-101 model. An image from the ILSVRC2012 validation set deformed to the labels of second highest (first row) and lowest (second row) probabilities (out of 1,000) for the original image. The vector fields and perturbations have been amplified for visualization.

Remarkably, PGD trained networks on MNIST are more resistant to adversarial deformations than ADef trained networks. However, for this result to be more conclusive, similar tests on ImageNet will have to be conducted. We wish to study this in future work.

REFERENCES

N. Akhtar and A. Mian. Threat of adversarial attacks on deep learning in computer vision: A survey. *IEEE Access*, 6:14410–14430, 2018. doi: 10.1109/ACCESS.2018.2807385.

Author Anonymous. Adversarial perturbations and deformations for convolutional neural networks, 2017.

Anish Athalye and Ilya Sutskever. Synthesizing robust adversarial examples. *arXiv preprint arXiv:1707.07397*, 2017.

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

Tom B Brown, Dandelion Mané, Aurko Roy, Martín Abadi, and Justin Gilmer. Adversarial patch. *arXiv preprint arXiv:1712.09665*, 2017.

Nicholas Carlini and David Wagner. Adversarial examples are not easily detected: Bypassing ten detection methods. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pp. 3–14. ACM, 2017a.

Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *IEEE Symposium on Security and Privacy (SP)*, pp. 39–57. IEEE, 2017b.

Logan Engstrom, Dimitris Tsipras, Ludwig Schmidt, and Aleksander Madry. A Rotation and a Translation Suffice: Fooling CNNs with Simple Transformations. *arXiv preprint arXiv:1712.02779*, 2017.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

Max Jaderberg, Karen Simonyan, Andrew Zisserman, and koray kavukcuoglu. Spatial transformer networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 28*, pp. 2017–2025. Curran Associates, Inc., 2015. URL `http://papers.nips.cc/paper/5854-spatial-transformer-networks.pdf`.

Harini Kannan, Alexey Kurakin, and Ian Goodfellow. Adversarial logit pairing. *arXiv preprint arXiv:1803.06373*, 2018.

Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial machine learning at scale. In *International Conference on Learning Representations*, 2017a.

Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world. In *International Conference on Learning Representations*, 2017b.

Yann LeCun. The MNIST database of handwritten digits. `http://yann.lecun.com/exdb/mnist/`.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=rJzIBfZAb`.

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)*, pp. 86–94, July 2017. doi: 10.1109/CVPR.2017.17.

Seyed Mohsen Moosavi Dezfooli, Alhussein Fawzi, and Pascal 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.

Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, pp. 506–519. ACM, 2017.

Aditi Raghunathan, Jacob Steinhardt, and Percy Liang. Certified defenses against adversarial examples. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=Bys4ob-Rb`.

Andras Rozsa, Ethan M Rudd, and Terrance E Boult. Adversarial diversity and hard positive generation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 25–32, 2016.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

Mahmood Sharif, Sruti Bhagavatula, Lujo Bauer, and Michael 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*, pp. 1528–1540. ACM, 2016.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2818–2826, 2016.

Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. Ensemble adversarial training: Attacks and defenses. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=rkZvSe-RZ.

Eric Wong and Zico Kolter. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *International Conference on Machine Learning*, 2018.

Chaowei Xiao, Jun-Yan Zhu, Bo Li, Warren He, Mingyan Liu, and Dawn Song. Spatially transformed adversarial examples. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=HyydRMZC-.

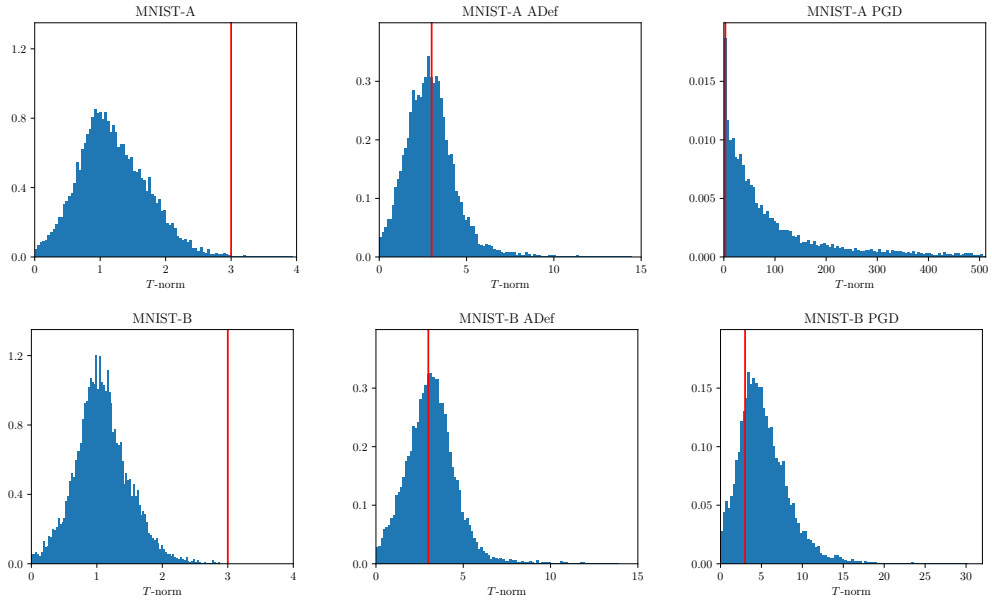

Figure 5: The (normalized) distribution of $\|\tau^*\|_T$ from the MNIST experiments. Deformations that fall to the left of the vertical line at $\varepsilon = 3$ are considered successful. The networks in the first column were trained using the original MNIST data, and the networks in the second and third columns were adversarially trained using ADef and PGD, respectively.

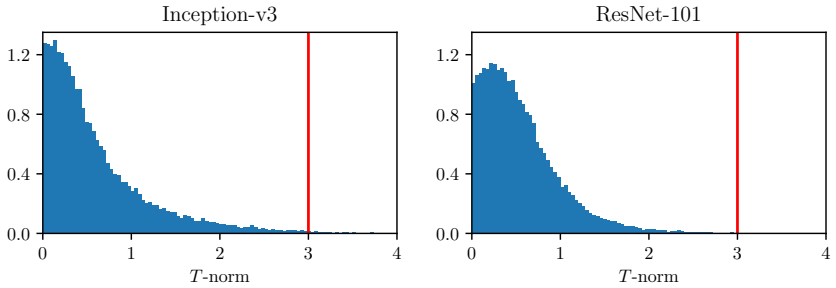

Figure 6: The (normalized) distribution of $\|\tau^*\|_T$ from the ImageNet experiments. Deformations that fall to the left of the vertical line at $\varepsilon = 3$ are considered successful.

A Distribution of vector field norms

Figures 5 and 6 show the distribution of the norms of the total deforming vector fields, τ^*, from the experiments in section 3. For networks that have not been adversarially trained, most deformations fall well below the threshold of $\epsilon = 3$. Out of the adversarially trained networks, only MNIST-A trained against PGD is truly robust against ADef. Further, a comparison between the first column of figure 5 and figure 6 indicates that ImageNet is much more vulnerable to adversarial deformations than MNIST, also considering the much higher resolution of the images in ImageNet. Thus, it would be very interesting to study the performance of ADef with adversarially trained network for ImageNet, as mentioned in the Conclusion.

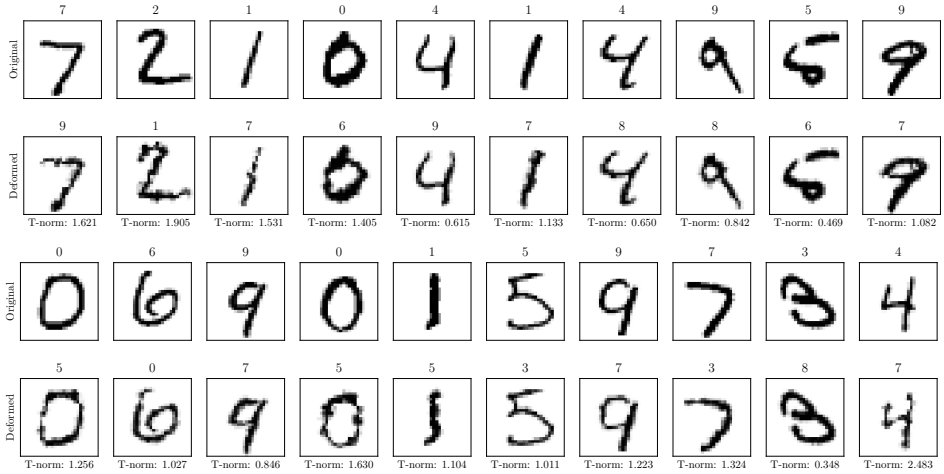

Figure 7: Adversarial deformations for MNIST-A. **First and third rows:** Original images from the MNIST test set. **Second and fourth rows:** The deformed images and the norms of the corresponding deforming vector fields.

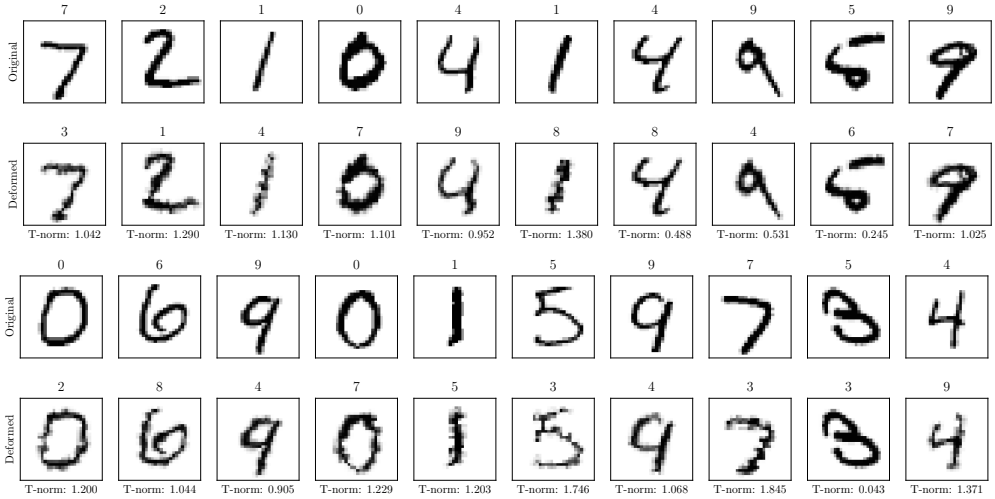

Figure 8: Adversarial deformations for MNIST-B. Note that image 9 in row 3 is misclassified, and is then deformed to its correct label.

B ADDITIONAL DEFORMED IMAGES

B.1 MNIST

Figures 7 and 8 show adversarial deformations for the models MNIST-A and MNIST-B, respectively. The attacks are performed using the same configuration as in the experiments in section 3.

B.2 IMAGENET

Figures 9 – 13 show additional deformed images resulting from attacking the Inception-v3 model using the same configuration as in the experiments in section 3. Similarly, figures 14 – 18 show deformed images resulting from attacking the ResNet-10 model. However, in order to increase variability in the output labels, we perform a targeted attack, targeting the label of 50th highest probability.

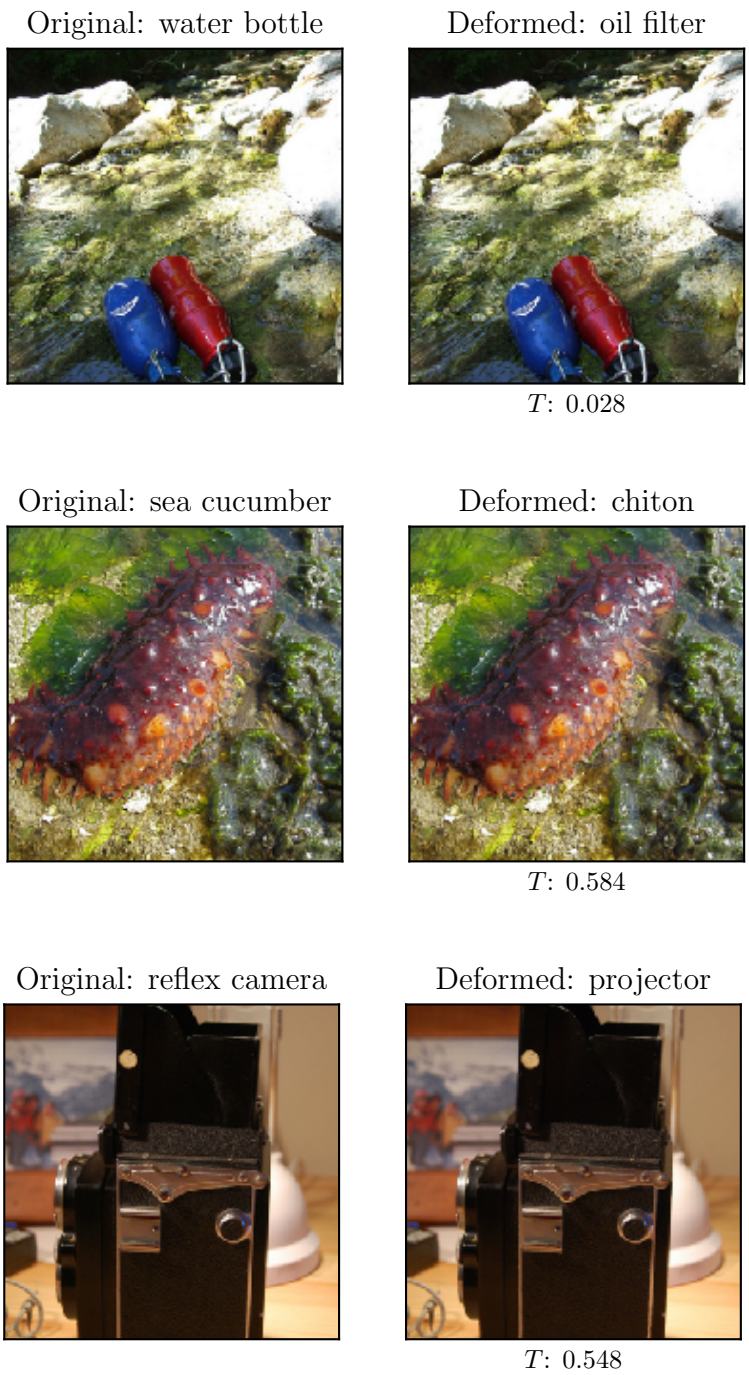

Figure 9: ADef attacks on the Inception-v3 model using the same configuration as in the experiments in section 3.

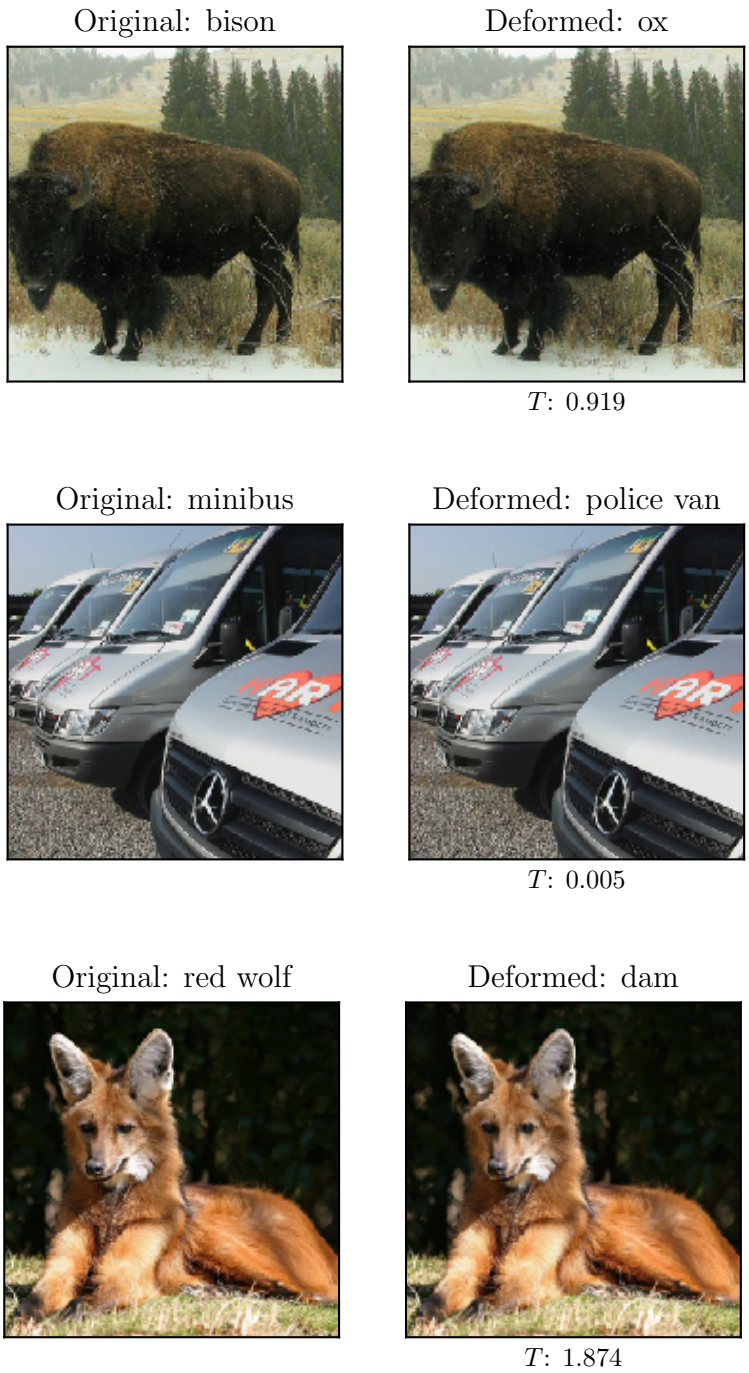

Figure 10: ADef attacks on the Inception-v3 model using the same configuration as in the experiments in section 3.

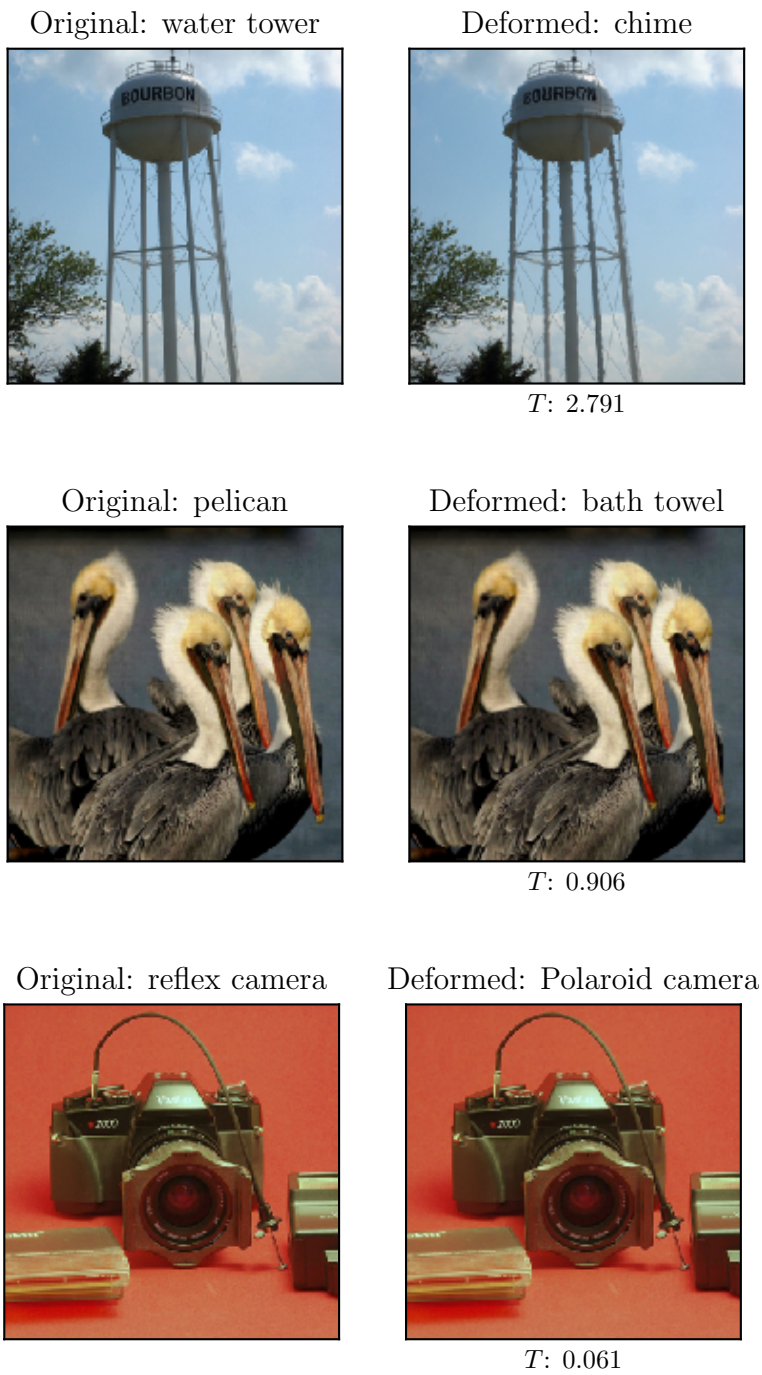

Figure 11: ADef attacks on the Inception-v3 model using the same configuration as in the experiments in section 3.

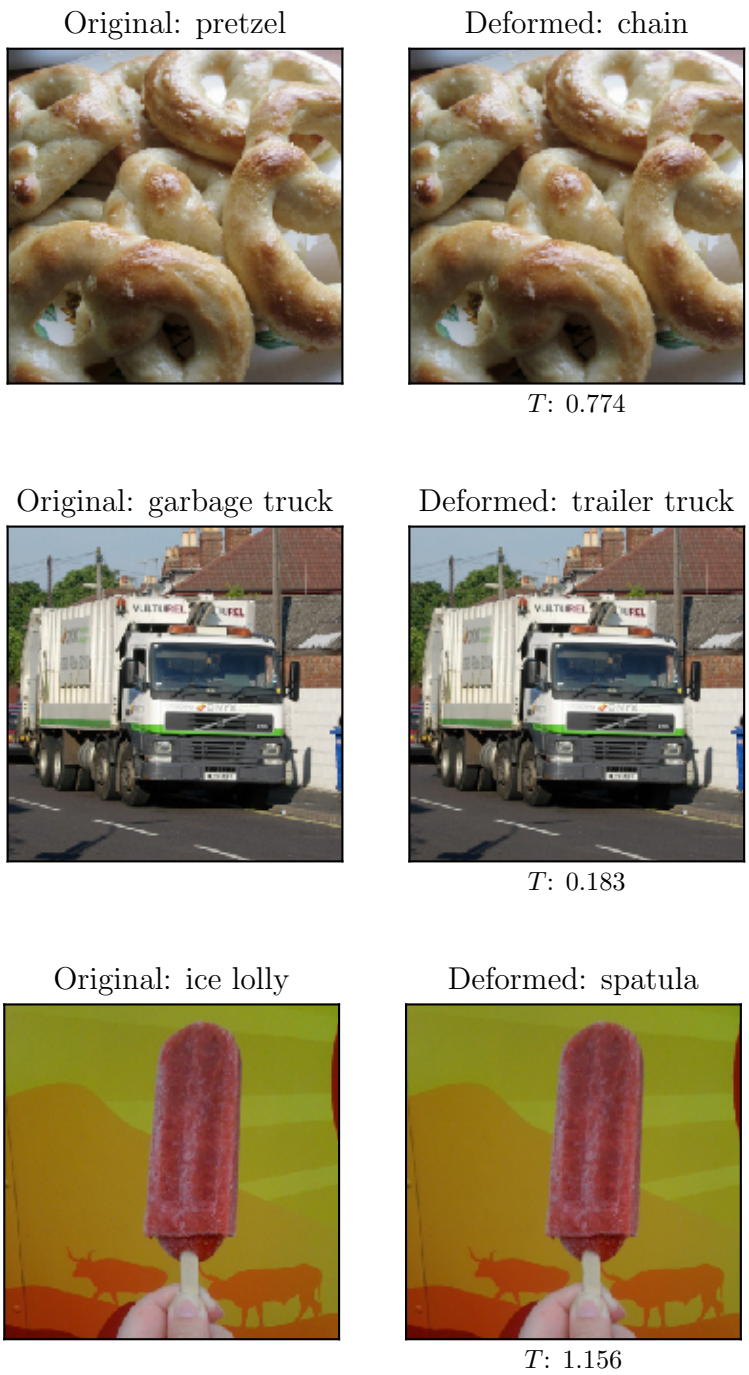

Figure 12: ADef attacks on the Inception-v3 model using the same configuration as in the experiments in section 3.

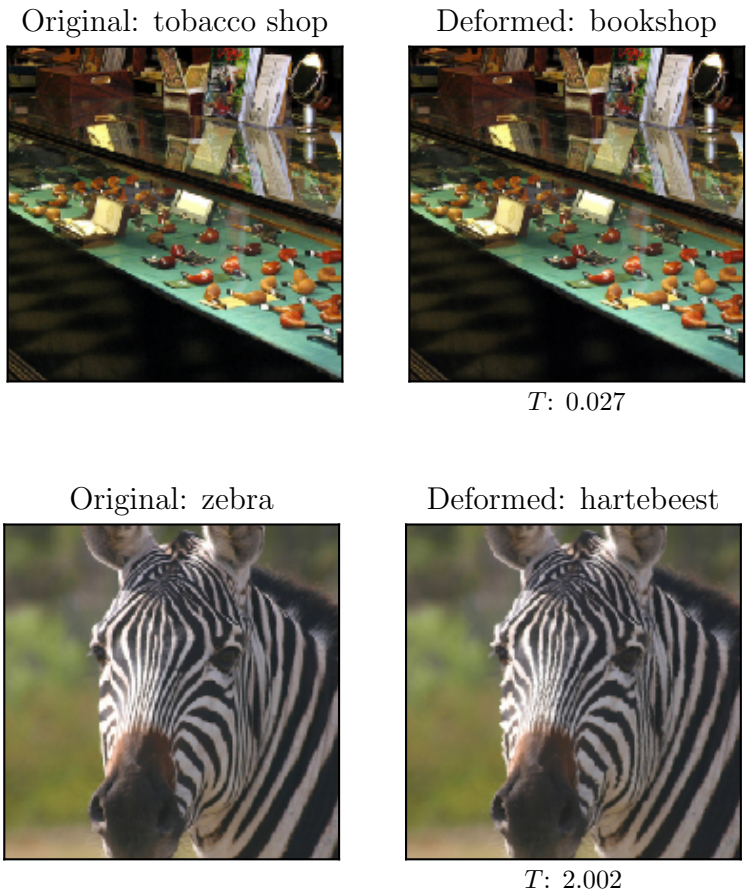

Figure 13: ADef attacks on the Inception-v3 model using the same configuration as in the experiments in section 3.

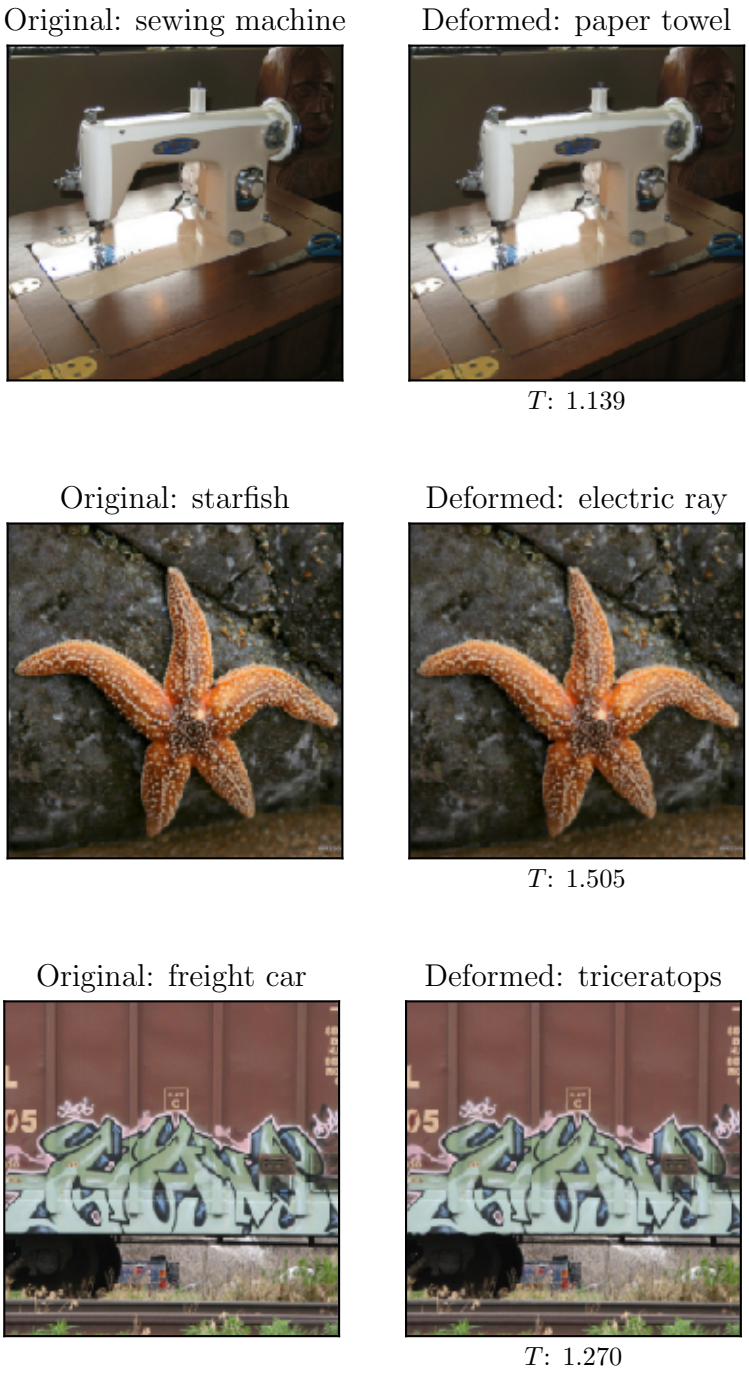

Figure 14: ADef attacks on the ResNet-101 model targeting the 50th most likely label.

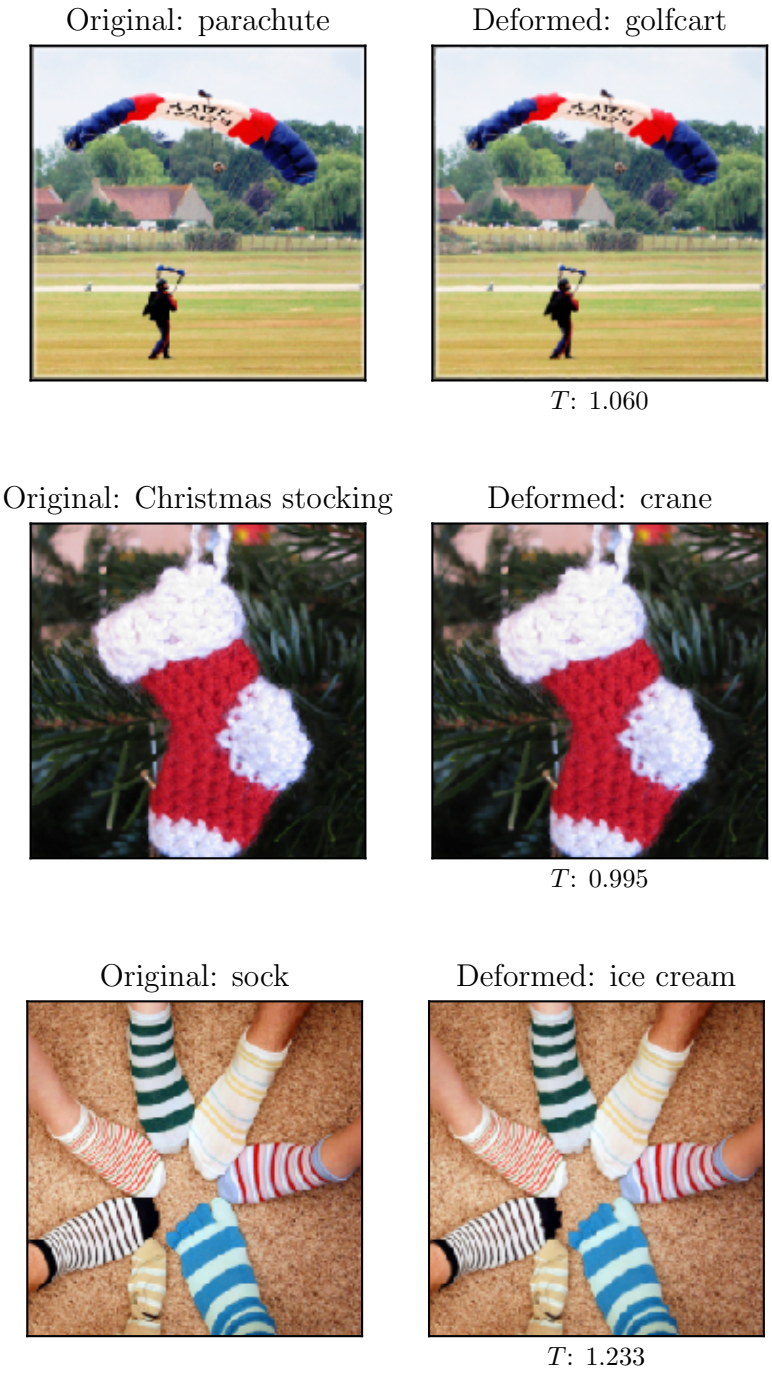

Figure 15: ADef attacks on the ResNet-101 model targeting the 50th most likely label.

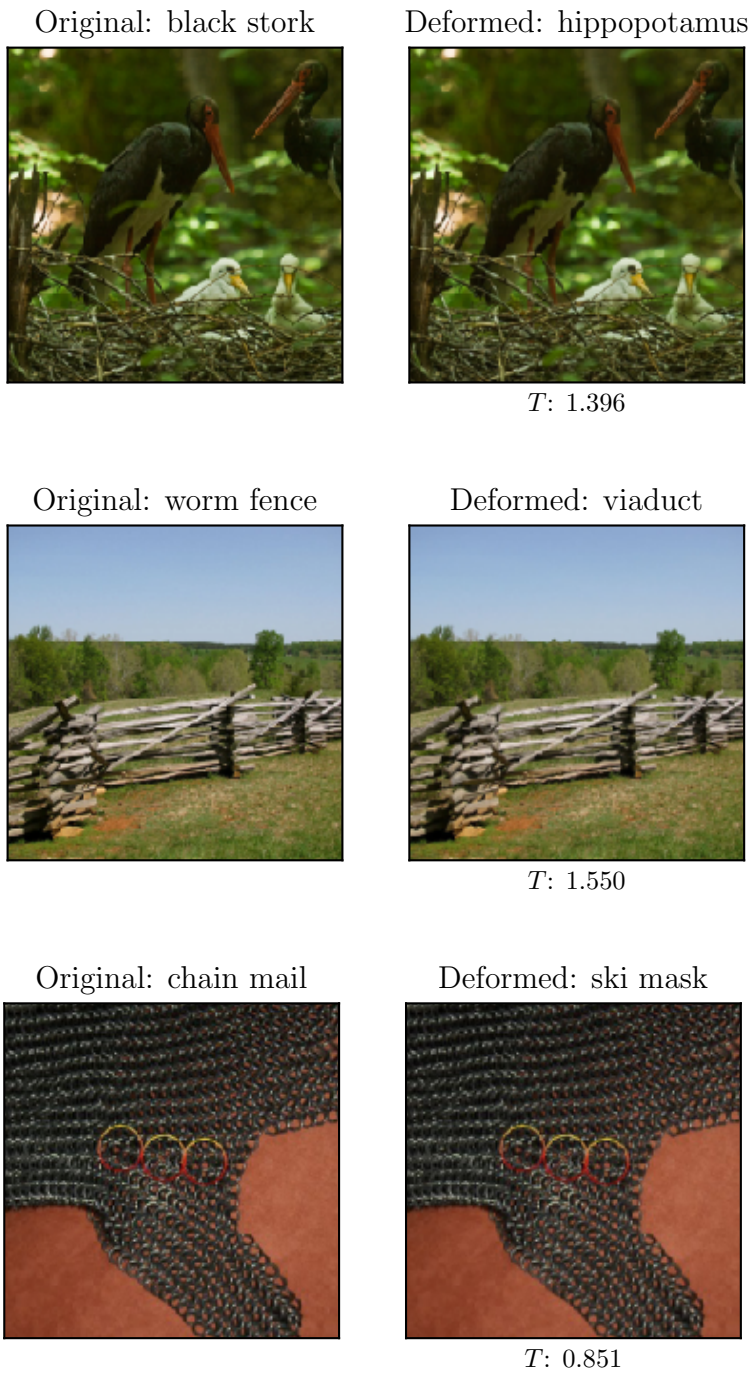

Figure 16: ADef attacks on the ResNet-101 model targeting the 50th most likely label.

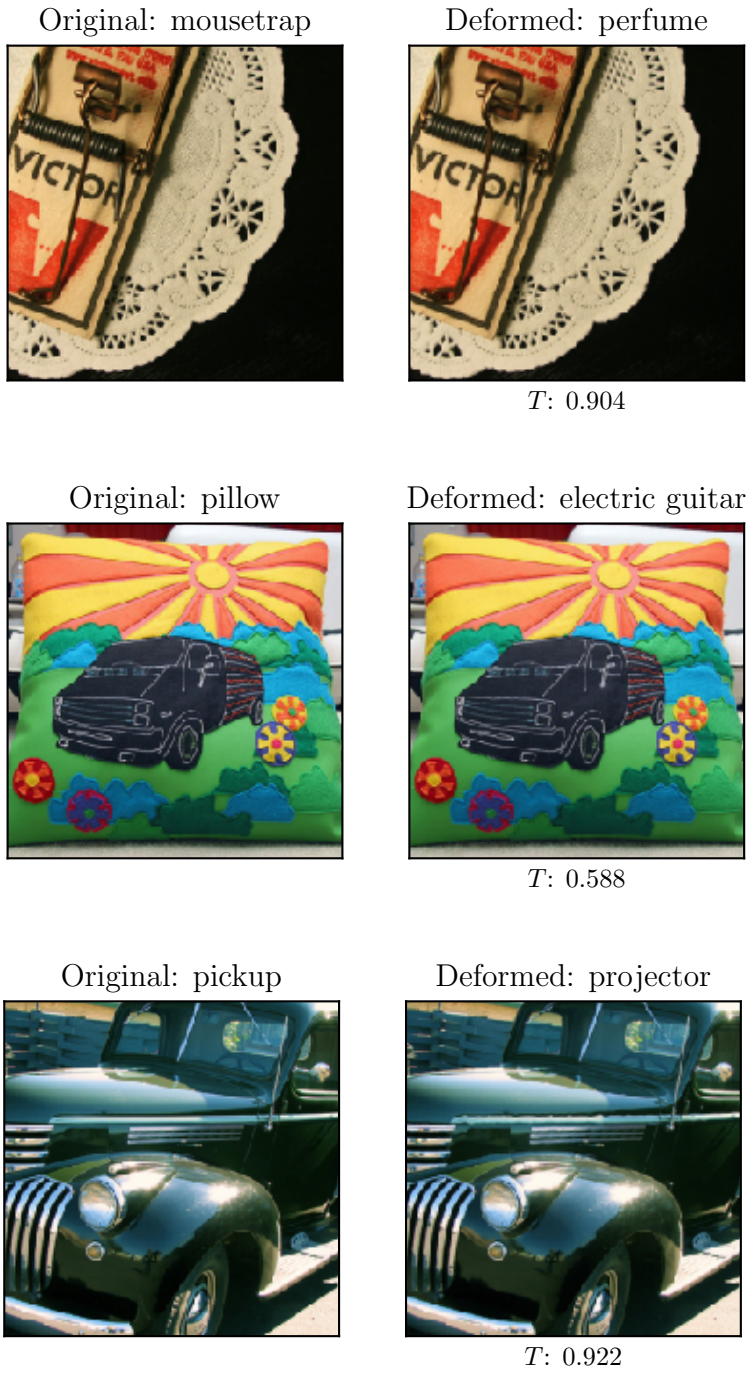

Figure 17: ADef attacks on the ResNet-101 model targeting the 50th most likely label.

Figure 18: ADef attacks on the ResNet-101 model targeting the 50th most likely label.

C MATHEMATICAL DERIVATION

To maintain anonymity during the reviewing process, we wish to include the section of the master's thesis Anonymous (2017) which includes the mathematical derivation of the ADef algorithm. This section can be removed altogether when the anonymity of this reference has been lifted. The remainder of the text is an anonymized version of the relevant part of Anonymous (2017). Keep in mind that the mathematical notation used in the following differs from the notation used in the main body of this paper.

C.1 ADVERSARIAL DEFORMATIONS

In this section, we model images (or signals in general) as functions in a L^2-Bochner space. Let $D := (0,1)^d$ be the domain on which our signals are defined. In the case of images we have $d = 2$, but we allow any dimension ≥ 1. Let c be the number of channels in the signal ($c = 1$ for grayscale images and $c = 3$ for color images). Define the space of signals to be the Bochner space $X := L^2(D, \mathbb{R}^c)$. It is a Hilbert space with the inner product

$$\langle \mathbf{x}, \mathbf{y} \rangle_X = \int_D \langle \mathbf{x}(\mathbf{p}), \mathbf{y}(\mathbf{p}) \rangle_{\mathbb{R}^c} \, \mathrm{d}\mu(\mathbf{p}) = \sum_{i=1}^{c} \int_D \mathbf{x}_i(\mathbf{p}) \mathbf{y}_i(\mathbf{p}) \, \mathrm{d}\mu(\mathbf{p})$$

where μ is the Lebesgue measure on D.

For all $\mathbf{x} \in X$, define

$$\|\mathbf{x}\|_{X,\infty} := \max_{i=1,\dots,c} \|\mathbf{x}_i\|_{L^\infty(D)} \, .$$

Note that this quantity is infinite for some $\mathbf{x} \in X$ and thus does not define a norm on X. In the space X, the box of valid signals is

$$\mathcal{B} = \{\mathbf{x} \in X : \ \|\mathbf{x}\|_{X,\infty} \leq 1\}.$$

In the literature, adversarial perturbations are usually regarded as "imperceptible" if they are small in some ℓ^p-norm. These norms, however, do not sufficiently describe what is imperceptable to the human eye. For example, if $\mathbf{x} \in \mathcal{B}$ and $\mathbf{u} \in \mathbb{R}^d$ is small, then the translated image $\mathbf{x}_\mathbf{u}$, defined by

$$\mathbf{x}_\mathbf{u}(\mathbf{p}) = \begin{cases} \mathbf{x}(\mathbf{p} + \mathbf{u}) & \text{if } \mathbf{p} + \mathbf{u} \in D \\ \mathbf{x}(\mathbf{p}) & \text{otherwise,} \end{cases}$$

is likely to be indistinguishable from $\mathbf{x}$ to the human eye. The corresponding perturbation $\mathbf{x}_\mathbf{u} - \mathbf{x}$ may however have a relatively large L^p-norm. Translation by $\mathbf{u}$ can be seen as a deformation of $\mathbf{x}$ by a constant vector field $\tau = \mathbf{u}$. In this section we generalize this example to deformations by arbitrary vector fields.

We search for *adversarial deformations* using the same strategy as in the search for adversarial perturbations: We assume that there exists a small deformation of $\mathbf{x}$ that changes its label and use a first order approximation of the classifier to find the corresponding vector field.

We consider deformations to be small when the corresponding vector field is small in the supremum norm. This allows us to bypass some technical difficulties that appear in the analysis with other L^p-norms. It may be of interest to limit the search for adversarial deformations to smooth vector fields. This can be achieved by limiting the search to the range of a Gaussian smoothing operator.

C.1.1 DEFORMATIONS

We now make precise the notion of deformations through vector fields on D. First, we specify a set U of *admissible* vector fields, and then for any bounded $\mathbf{x} \in X$ we define a family of deformed signals $(\mathbf{x}_\tau)_{\tau \in U}$.

Let $0 < \delta < \frac{1}{2}$, $K := [\delta, 1-\delta]^d \subseteq D$ and denote by $T := L^\infty(K, \mathbb{R}^d)$ the Bochner space of vector fields $\tau : \mathbf{p} \mapsto \tau_\mathbf{p}$ supported on K and bounded with respect to the norm $\|\tau\|_T = \operatorname{ess\,sup}_{\mathbf{p} \in K} \|\tau_\mathbf{p}\|_{\mathbb{R}^d}$. For a vector field τ, define the deformation

$$\theta_\tau : D \to \mathbb{R}^d, \quad \mathbf{p} \mapsto \mathbf{p} + \tau_\mathbf{p}$$

(taking $\tau_{\mathbf{p}} = \mathbf{0}$ for $\mathbf{p} \notin K$) and define the set of admissible vector fields as

$$U = \{\tau \in T : \ \theta_\tau(D) \subseteq D\}.$$

Note that since

$$\inf \left\{ \|\mathbf{p} - \mathbf{q}\|_{\mathbb{R}^d} : \ \mathbf{p} \in K, \ \mathbf{q} \in \mathbb{R}^d \setminus D \right\} = \delta,$$

we have

$$B_\delta(0) := \{\tau \in T : \ \|\tau\|_T < \delta\} \subseteq U.$$

For any $\mathbf{x} \in X$ with $\|\mathbf{x}\|_{X,\infty} < \infty$ we define the *deformation operator* of $\mathbf{x}$:

$$\Theta_{\mathbf{x}} : U \to X, \quad \tau \mapsto \mathbf{x}_\tau$$

where $\mathbf{x}_\tau := \mathbf{x} \circ \theta_\tau$. Observe that $\|\mathbf{x}_\tau\|_{X,\infty} \leq \|\mathbf{x}\|_{X,\infty} < \infty$ for any $\tau \in U$. Hence, $\|\mathbf{x}_\tau\|_X < \infty$ for all $\tau \in U$ and $\Theta_{\mathbf{x}}$ is well defined.

Lemma C.1. *If $\mathbf{x}$ is twice continuously differentiable in a neighborhood of K and $\|\mathbf{x}\|_X < \infty$, then $\Theta_{\mathbf{x}} : U \to X$ is Fréchet differentiable at 0 and its Fréchet derivative at 0, $\mathcal{D}_0\Theta_{\mathbf{x}}$, is equal to the operator*

$$A : T \to X, \quad \tau \mapsto \chi_\tau,$$

where

$$\chi_\tau : D \to \mathbb{R}^c, \quad \mathbf{p} \mapsto (\mathrm{D}_{\mathbf{p}}\mathbf{x})\, \tau_{\mathbf{p}}.$$

Proof. First we note that any $\tau \in T$ is bounded and supported on K and since $\mathbf{x}$ has continuous derivatives on K, we get

$$\begin{aligned}
\|\chi_\tau\|_X^2 &= \int_D \|(\mathrm{D}_{\mathbf{p}}\mathbf{x})\, \tau_{\mathbf{p}}\|_{\mathbb{R}^c}^2 \, \mathrm{d}\mu(\mathbf{p}) \\
&\leq \int_K \|\mathrm{D}_{\mathbf{p}}\mathbf{x}\|_{\mathrm{op}}^2 \, \|\tau_{\mathbf{p}}\|_{\mathbb{R}^d}^2 \, \mathrm{d}\mu(\mathbf{p}) \\
&\leq \mu(K) \left(\sup_{\mathbf{p} \in K} \|\mathrm{D}_{\mathbf{p}}\mathbf{x}\|_{\mathrm{op}}^2 \right) \|\tau\|_T^2 < \infty,
\end{aligned}$$

where

$$\|\mathrm{D}_{\mathbf{p}}\mathbf{x}\|_{\mathrm{op}} = \sup_{\mathbf{u} \in \mathbb{R}^d \setminus \{\mathbf{0}\}} \frac{\|(\mathrm{D}_{\mathbf{p}}\mathbf{x})\, \mathbf{u}\|_{\mathbb{R}^c}}{\|\mathbf{u}\|_{\mathbb{R}^d}}$$

is the operator norm of $\mathrm{D}_{\mathbf{p}}\mathbf{x}$. Hence, the mapping

$$A : T \to X, \quad \tau \mapsto \chi_\tau$$

is a well defined bounded linear operator.

To understand $\Theta_{\mathbf{x}}$, we consider local linear approximations of $\mathbf{x}$. For any $\mathbf{p} \in K$ and $\mathbf{u} \in \mathbb{R}^d$ small enough we have

$$\mathbf{x}(\mathbf{p} + \mathbf{u}) = \mathbf{x}(\mathbf{p}) + (\mathrm{D}_{\mathbf{p}}\mathbf{x})\, \mathbf{u} + \eta_{\mathbf{p}}(\mathbf{u})$$

where $\|\eta_{\mathbf{p}}(\mathbf{u})\|_{\mathbb{R}^c} \|\mathbf{u}\|_{\mathbb{R}^d}^{-2} \to 0$ as $\mathbf{u} \to \mathbf{0}$. Moreover, by continuity of the second derivatives of $\mathbf{x}$ on the compact set K, we have the uniform bound $\|\eta_{\mathbf{p}}(\mathbf{u})\|_{\mathbb{R}^c} \leq C \|\mathbf{u}\|_{\mathbb{R}^d}^2$ for all $\mathbf{p} \in K$ and all $\mathbf{u}$ with $\|\mathbf{u}\|_{\mathbb{R}^d} < \delta$ for some constant $C > 0$.

Note that $\theta_0 = \mathrm{id}_D$, so $\mathbf{x}_0 = \mathbf{x}$. Hence for $\tau \in U$ we have for all $\mathbf{p} \in K$

$$\begin{aligned}
\mathbf{x}_\tau(\mathbf{p}) - \mathbf{x}_0(\mathbf{p}) &= \mathbf{x}(\mathbf{p} + \tau_{\mathbf{p}}) - \mathbf{x}(\mathbf{p}) \\
&= (\mathrm{D}_{\mathbf{p}}\mathbf{x})\, (\tau_{\mathbf{p}}) + \eta_{\mathbf{p}}(\tau_{\mathbf{p}}) \\
&= \chi_\tau(\mathbf{p}) + \eta_{\mathbf{p}}(\tau_{\mathbf{p}})
\end{aligned}$$

and for $\mathbf{p} \notin K$ we have $\mathbf{x}_\tau(\mathbf{p}) - \mathbf{x}_0(\mathbf{p}) = \chi_\tau(\mathbf{p}) = \mathbf{0}$. Recall that U, the set of admissible vector fields, contains an open ball around $0 \in T$, so the following limit is well defined.

$$\begin{aligned}
\lim_{\tau \to 0} \frac{\|\Theta_{\mathbf{x}}(\tau) - \Theta_{\mathbf{x}}(0) - A(\tau)\|_X}{\|\tau\|_T} &= \lim_{\tau \to 0} \frac{1}{\|\tau\|_T} \left(\int_K \|\eta_{\mathbf{p}}(\tau_{\mathbf{p}})\|^2_{\mathbb{R}^c} \, \mathrm{d}\mu(\mathbf{p}) \right)^{1/2} \\
&\leq \lim_{\tau \to 0} \frac{1}{\|\tau\|_T} \left(\int_K C^2 \, \|\tau_{\mathbf{p}}\|^4_{\mathbb{R}^d} \, \mathrm{d}\mu(\mathbf{p}) \right)^{1/2} \\
&\leq \lim_{\tau \to 0} \frac{1}{\|\tau\|_T} \left(\mu(K) C^2 \, \|\tau\|^4_T \right)^{1/2} \\
&\leq \lim_{\tau \to 0} \left(C \sqrt{\mu(K)} \, \|\tau\|_T \right) = 0.
\end{aligned}$$

This proves that $\Theta_{\mathbf{x}}$ is Fréchet differentiable and that $\mathcal{D}_0 \Theta_{\mathbf{x}} = A$. □

Remark C.2. Alternatively, we could have considered vector fields in the space $\overline{T} := L^2(K, \mathbb{R}^d)$ instead of T. Note that while $T \subseteq \overline{T}$, the set of admissible vector fields

$$\overline{U} := \{\tau \in \overline{T} : \ \theta_\tau(D) \subseteq D\}^2$$

does not contain an open ball around 0 in $\overline{T}$ and the Fréchet derivative of $\Theta_{\mathbf{x}}$ at 0 is not well defined. To see this, let $\sigma > 0$ and consider the vector field $\tau^\sigma : K \to \mathbb{R}^d$, defined for all $\mathbf{p} \in K$ by

$$\tau^\sigma_{\mathbf{p}} = \begin{cases} \mathbf{0} & \text{if } \|\mathbf{p} - \mathbf{q}\|_{\mathbb{R}^d} \geq \sigma \\ \mathbf{e}_1 & \text{otherwise,} \end{cases}$$

where $\mathbf{q} = (\frac{1}{2}, \ldots, \frac{1}{2}) \in \mathbb{R}^d$ is the center of the hypercube K and $\mathbf{e}_1 = (1, 0, \ldots, 0)$ is the first standard basis vector in $\mathbb{R}^d$. The vector field τ^σ is not admissible since $\mathbf{q} + \mathbf{e}_1 = (\frac{3}{2}, \frac{1}{2}, \ldots, \frac{1}{2}) \notin D$ and $B_\sigma(\mathbf{q} + \mathbf{e}_1) \subseteq \theta_{\tau^\sigma}(D)$. Although $\|\tau^\sigma\|_T = 1$ for all $\sigma > 0$, we have

$$\|\tau^\sigma\|_{\overline{T}} = \int_K \left\|\tau^\sigma_{\mathbf{p}}\right\|^2_{\mathbb{R}^d} \mathrm{d}\mu(\mathbf{p}) = \int_{B_\sigma(\mathbf{q})} \|\mathbf{e}_1\|^2_{\mathbb{R}^d} \, \mathrm{d}\mu(\mathbf{p}) = \mu(B_\sigma(\mathbf{q})) = \sigma^d V_d,$$

where V_d stands for the volume of the d-dimensional unit ball. Hence, for any $\varepsilon > 0$, the non-admissible vector field τ^σ lies in $B_\varepsilon(0)$ as long as $\sigma < (\varepsilon / V_d)^{1/d}$.

The fact that $\Theta_{\mathbf{x}}$ is not defined on a whole neighborhood of 0 in $\overline{T}$ would lead us to consider the Gâteaux derivative of $\Theta_{\mathbf{x}}$ at $0 \in \overline{T}$ in any direction τ in the subspace T:

$$\mathrm{D}^G{}_0 \Theta_{\mathbf{x}}(\tau) = \lim_{\varepsilon \to 0} \frac{\Theta_{\mathbf{x}}(\varepsilon \tau) - \Theta_{\mathbf{x}}(0)}{\varepsilon}$$

For the sake of simplicity, we stick to our choice $T = L^\infty(K, \mathbb{R}^d)$.

C.1.2 Finding adversarial deformations

Suppose we have a classifier on the signal space into L categories,

$$F : X \to \mathbb{R}^L,$$

and an interpretable signal $\mathbf{x} \in \mathcal{B}$ classified as

$$l := \operatorname*{arg\,max}_{k=1,\ldots,L} F_k(\mathbf{x}).$$

Further assume that $\mathbf{x}$ is twice continuously differentiable, and that F is Fréchet differentiable at $\mathbf{x}$. Our goal is to find a small deformation $\tau \in U$ such that

$$l \neq \operatorname*{arg\,max}_{k=1,\ldots,L} F_k(\mathbf{x}_\tau).$$

We proceed as with the DeepFool algorithm. First we target a different label $k \neq l$ and define the binary classifier $f_k := F_k - F_l$. The function $f_k : X \to \mathbb{R}$ returns positive values for signals for

[2] Since any vector field in $\overline{U}$ is bounded, we have $\overline{U} = U$.

which the classifier F prefers the label k over the label l and it returns negative values for signals for which F prefers l over k. Composing this new classifier with the deformation operator of $\mathbf{x}$ we get another binary classifier $g_k := f_k \circ \Theta_{\mathbf{x}}$ on U. We have $g_k(\tau) > 0$ for vector fields τ such that $\mathbf{x}_\tau$ is considered to belong to class k rather than class l. Likewise $g_k(\tau) < 0$ if τ is unsuccessful in changing the preference from l to k.

By assumption, f_k is Fréchet differentiable at $\mathbf{x}$. Its Fréchet derivative, $\mathcal{D}_{\mathbf{x}} f_k$, is a continuous linear functional on the Hilbert space X, and by the Riesz representation theorem there exists a unique function $\psi \in X$ such that

$$(\mathcal{D}_{\mathbf{x}} f_k)\mathbf{y} = \langle \psi, \mathbf{y} \rangle_X, \quad \text{for all } \mathbf{y} \in X.$$

By the chain rule and lemma C.1, g_k is differentiable at 0. Let us calculate the Fréchet derivative of g_k in the direction of $\tau \in T$. The first equality is due to $\Theta_{\mathbf{x}}(0) = \mathbf{x}$.

$$\begin{aligned}
(\mathcal{D}_0 g_k)\tau = (\mathcal{D}_{\mathbf{x}} f_k)(\mathcal{D}_0 \Theta_{\mathbf{x}})\tau &= \langle \psi, \chi_\tau \rangle_X \\
&= \sum_{i=1}^{c} \langle\, \psi_i,\ (\chi_\tau)_i \,\rangle_{L^2(D)} \\
&= \sum_{i=1}^{c} \langle\, \psi_i,\ \langle \nabla \mathbf{x}_i, \tau \rangle_{\mathbb{R}^d} \,\rangle_{L^2(D)} \\
&= \sum_{i=1}^{c} \langle\, \psi_i,\ \sum_{j=1}^{d} \tau_j \partial_j \mathbf{x}_i \,\rangle_{L^2(D)} \\
&= \sum_{i=1}^{c} \sum_{j=1}^{d} \langle\, \psi_i,\ \tau_j \partial_j \mathbf{x}_i \,\rangle_{L^2(D)} \\
&= \sum_{i=1}^{c} \sum_{j=1}^{d} \langle\, \psi_i \partial_j \mathbf{x}_i,\ \tau_j \,\rangle_{L^2(K)}. \\
&= \sum_{j=1}^{d} \langle\, \sum_{i=1}^{c} \psi_i \partial_j \mathbf{x}_i,\ \tau_j \,\rangle_{L^2(K)}.
\end{aligned} \tag{8}$$

The second to last step is due to the partial derivatives $\partial_j \mathbf{x}_i : \mathbb{R} \to \mathbb{R}$ being continuous and hence bounded on the compact set K and τ_j being zero outside of K:

$$\langle\, \psi_i,\ \tau_j \partial_j \mathbf{x}_i \,\rangle_{L^2(D)} = \int_D \psi_i \tau_j \partial_j \mathbf{x}_i = \int_K \psi_i \partial_j \mathbf{x}_i \tau_j = \langle\, \psi_i \partial_j \mathbf{x}_i,\ \tau_j \,\rangle_{L^2(K)}.$$

Note that $\tau_j \in L^2(K)$ since it is bounded on account of $\tau \in T = L^\infty(K, \mathbb{R}^d)$ and $\mu(K) < \infty$.

If there exists a vector field $\tau \in U$ such that

$$(\mathcal{D}_0 g_k)\, \tau = -g_k(0) \tag{9}$$

and if it is small enough, then by first order approximation

$$g_k(\tau) \approx g_k(0) + (\mathcal{D}_0 g_k)\, \tau = 0. \tag{10}$$

In this case $\mathbf{x}_\tau$ lies approximately on the decision boundary of f_k. For $j = 1, \ldots, d$ define $\alpha_j \in L^2(K)$ by setting

$$\alpha_j = \langle \psi, \partial_j \mathbf{x} \rangle_{\mathbb{R}^c} = \sum_{i=1}^{c} \psi_i \partial_j \mathbf{x}_i. \tag{11}$$

Then by the calculations above, the vector field

$$\tau(\mathbf{x}, k) := \frac{-g_k(0)}{\sum_{j=1}^{d} \|\alpha_j\|^2_{L^2(K)}} \begin{pmatrix} \alpha_1 \\ \vdots \\ \alpha_d \end{pmatrix}$$

satisfies condition (9), provided that it is bounded. Even if $\tau(\mathbf{x}, k)$ is bounded, it may not be admissible and small enough for (10) to be a good approximation. However, we still have the freedom to choose the targeted label k to minimize $\|\tau(\mathbf{x}, k)\|_T$.

We define an algorithm for finding adversarial deformations for $\mathbf{x}$: Initialize by $\mathbf{x}^0 := \mathbf{x}$. Then for $n \geq 0$ let

$$k^n := \arg\min_{k \neq l} \|\tau(\mathbf{x}^n, k)\|_T$$

and $\tau^n := \tau(\mathbf{x}^n, k^n)$ and define

$$\mathbf{x}^{n+1} := \Theta_{\mathbf{x}}^n(\tau^n)$$

where $\Theta_{\mathbf{x}}^n$ is the deformation operator of $\mathbf{x}^n$. Subtleties of this update step are addressed in section C.1.3.

Once $l \neq \arg\max_{k=1,\ldots,L} F_k(\mathbf{x}^n)$, the iteration stops and the algorithm outputs the deformed image

$$\hat{\mathbf{x}} := \mathbf{x}^n = \Theta_{\mathbf{x}}^0\left(\sum_{m=0}^{n-1} \tau^m\right).$$

One may also limit the number of iterations to some $n_{\max} \geq 1$ and the magnitude of the total deformation $\sum_{m=0}^{n-1} \tau^m$ (in some norm) to a number $t_{\max} > 0$.

C.1.3 Smooth deformations

In order to iterate the deformation process, we need to assume that F is also Fréchet differentiable at $\mathbf{x}_\tau$. Moreover, the function $\mathbf{x}_\tau = \Theta_{\mathbf{x}}(\tau) : D \to \mathbb{R}^c$ does not in general satisfy the conditions of lemma C.1. However, if $\mathbf{x}$ is twice continuously differentiable on an open set $E \subseteq D$ containing K, and τ is smooth on the interior of K (denoted by $\mathring{K}$), then $\mathbf{x}_\tau$ is twice continuously differentiable on the open set

$$\mathring{K} \cap \theta_\tau^{-1}(E) = \left\{\mathbf{p} \in \mathring{K} : \ \mathbf{p} + \tau_{\mathbf{p}} \in E\right\}.$$

If $\|\tau\|_T$ is sufficiently small, then there exists a number $\tilde{\delta}$ with $0 < \delta < \tilde{\delta} < 1/2$ such that

$$\tilde{K} := [\tilde{\delta}, 1 - \tilde{\delta}]^d \subseteq \mathring{K} \cap \theta_\tau^{-1}(E).$$

Thus $\mathbf{x}_\tau$ satisfies the conditions of lemma C.1 with K replaced by the smaller domain $\tilde{K}$, and we can repeat the deformation step on this shrunk set. In this section, we give a simple method for obtaining a smooth vector field τ that satisfies condition (9). This way, one can iterate the deformation without worry, given that F is everywhere Fréchet differentiable and the domain K is allowed to shrink with each iteration.

Let $\varphi : \mathbb{R}^d \to \mathbb{R}$ be a smooth, integrable function with the property $\varphi(\mathbf{p}) = \varphi(-\mathbf{p})$ for all $\mathbf{p} \in \mathbb{R}^d$. For example, φ may be chosen to be the d-dimensional Gaussian function of width $\sigma > 0$,

$$\varphi(\mathbf{p}) = \frac{1}{(2\pi\sigma^2)^{d/2}} \exp\left(-\frac{1}{2\sigma^2}\|\mathbf{p}\|_2^2\right) \quad \text{for all } \mathbf{p} \in \mathbb{R}^d.$$

Define the linear operator $\mathcal{S} : T \to T$ by

$$\mathcal{S}\tau := \begin{pmatrix} (\varphi * \tau_1)_{|K} \\ \vdots \\ (\varphi * \tau_d)_{|K} \end{pmatrix} \quad \text{for all } \tau \in T,$$

where $(\varphi * \tau_j)_{|K}$ denotes the restriction of $\varphi * \tau_j$ to K and the convolution is defined by extending τ_j by zero outside of K. The operator is well defined since for each $j = 1, \ldots, d$, the function $\varphi * \tau_j$ is smooth, K is compact, and thus $(\mathcal{S}\tau)_j = (\varphi * \tau_j)_{|K}$ is bounded. Any vector field in the range of $\mathcal{S}$ is smooth on the interior of K, and hence we shall search for a vector field $\tau \in T$ such that $\tilde{\tau} := \mathcal{S}\tau$ satisfies condition (9).

Note that for any $u, v \in L^2(K)$, we have

$$\langle\, u,\ (\varphi * v)_{|K} \,\rangle_{L^2(K)} = \langle\, (\varphi * u)_{|K},\ v \,\rangle_{L^2(K)}$$

since $\varphi = \varphi(-\cdot)$. By calculation (8), we have for all $\tau \in T$:

$$(\mathcal{D}_0 g_k)(\mathcal{S}\tau) = \sum_{j=1}^{d} \langle\, \alpha_j,\ (\varphi * \tau_j)_{|_K} \,\rangle_{L^2(K)} = \sum_{j=1}^{d} \langle\, (\varphi * \alpha_j)_{|_K},\ \tau_j \,\rangle_{L^2(K)},$$

where $\alpha_1, \ldots, \alpha_d$ are defined by formula (11). Set $\tilde{\alpha}_j := (\varphi * \alpha_j)_{|_K}$ for $j = 1, \ldots, d$ and

$$\tau := \frac{-g_k(0)}{\sum_{j=1}^{d} \|\tilde{\alpha}_j\|^2_{L^2(K)}} \begin{pmatrix} \tilde{\alpha}_1 \\ \vdots \\ \tilde{\alpha}_d \end{pmatrix}.$$

Then $\tilde{\tau} := \mathcal{S}\tau$ is a smooth vector field on K that satisfies condition (9),

$$(\mathcal{D}_0 g_k)\tilde{\tau} = -g_k(0).$$

Hence we may substitute $\tilde{\tau}$ for $\tau(\mathbf{x}, k)$ in the algorithm described in section C.1.2.

C.1.4 Adversarial deformations in practice

We now make explicit the discrete version of the deformation algorithm. Specifically, we consider square images $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3) \in X = \mathbb{R}^{3\times w\times w}$ of $P = w^2$ pixels and $c = 3$ color channels. Here it is important to maintain the structure of the image $\mathbf{x}$ as a triplet of square matrices instead of representing it simply as a column vector in $\mathbb{R}^{3P}$. Let $F : X \to \mathbb{R}^L$ be a differentiable classifier and let $\mathbf{x}^0 \in X$ be an image with label $l = \arg\max_k F_k(\mathbf{x}^0)$.

Now let $n \geq 0$ be an integer and write $\mathbf{x} = \mathbf{x}^n$ for simplicity. The image $\mathbf{x} \in X$ can be thought of as a sampling of a twice differentiable function $\xi : (0,1)^2 \to \mathbb{R}^3$ through

$$\mathbf{x}_{i,p,q} = \xi_i\left(\frac{p}{w+1}, \frac{q}{w+1}\right), \quad \text{for } p, q = 1, \ldots, w, \text{ and } i = 1, 2, 3.$$

We approximate the partial derivatives of ξ by finite differences. For $i = 1, 2, 3$, define the matrices $\mathrm{D}_1\mathbf{x}_i, \mathrm{D}_2\mathbf{x}_i \in \mathbb{R}^{w\times w}$ by central differences where possible

$$[\mathrm{D}_1\mathbf{x}_i]_{p,q} = \frac{w+1}{2}\left(\mathbf{x}_{i,p+1,q} - \mathbf{x}_{i,p-1,q}\right) \quad \text{for } p = 2, \ldots, (w-1) \text{ and all } q,$$

$$[\mathrm{D}_2\mathbf{x}_i]_{p,q} = \frac{w+1}{2}\left(\mathbf{x}_{i,p,q+1} - \mathbf{x}_{i,p,q-1}\right) \quad \text{for } q = 2, \ldots, (w-1) \text{ and all } p,$$

and define the remaining entries on the boundary by forward and backward differences accordingly:

$$\begin{aligned}
[\mathrm{D}_1\mathbf{x}_i]_{1,q} &= (w+1)\left(\mathbf{x}_{i,2,q} - \mathbf{x}_{i,1,q}\right) \quad \text{and} \\
[\mathrm{D}_1\mathbf{x}_i]_{w,q} &= (w+1)\left(\mathbf{x}_{i,w,q} - \mathbf{x}_{i,w-1,q}\right) \quad \text{for all } q, \\
[\mathrm{D}_2\mathbf{x}_i]_{p,1} &= (w+1)\left(\mathbf{x}_{i,p,2} - \mathbf{x}_{i,p,1}\right) \quad \text{and} \\
[\mathrm{D}_2\mathbf{x}_i]_{p,w} &= (w+1)\left(\mathbf{x}_{i,p,w} - \mathbf{x}_{i,p,w-1}\right) \quad \text{for all } p.
\end{aligned}$$

The entries of $\mathrm{D}_1\mathbf{x}_i$ and $\mathrm{D}_2\mathbf{x}_i$ then approximate the partial derivatives of ξ_i:

$$\begin{aligned}
\partial_1\xi_i\left(\frac{p}{w+1}, \frac{q}{w+1}\right) &\approx [\mathrm{D}_1\mathbf{x}_i]_{p,q} \quad \text{and} \\
\partial_2\xi_i\left(\frac{p}{w+1}, \frac{q}{w+1}\right) &\approx [\mathrm{D}_2\mathbf{x}_i]_{p,q}
\end{aligned}$$

for $p, q = 1, \ldots, w$ and $i = 1, 2, 3$.

Let $k \neq l$ be the target label. By assumption, the gradient $\nabla_{\mathbf{x}} F_k = ((\nabla_{\mathbf{x}} F_k)_1, (\nabla_{\mathbf{x}} F_k)_2, (\nabla_{\mathbf{x}} F_k)_3) \in \mathbb{R}^{3\times w\times w}$ is well defined. Define the matrices $\alpha_1, \alpha_2 \in \mathbb{R}^{w\times w}$ by inner product over the color dimension as in (11),

$$\begin{aligned}
\alpha_1 &= \sum_{i=1}^{3} \left(\nabla_{\mathbf{x}} F_k - \nabla_{\mathbf{x}} F_l\right)_i \odot \mathrm{D}_1\mathbf{x}_i, \\
\alpha_2 &= \sum_{i=1}^{3} \left(\nabla_{\mathbf{x}} F_k - \nabla_{\mathbf{x}} F_l\right)_i \odot \mathrm{D}_2\mathbf{x}_i.
\end{aligned} \tag{12}$$

where $\odot$ denotes the Hadamard (entry-wise) product of matrices. The matrices α_1 and α_2 are the (scaled) components of the discrete vector field

$$\tau(\mathbf{x}, k) := \frac{F_l(\mathbf{x}) - F_k(\mathbf{x})}{\|\alpha_1\|_F^2 + \|\alpha_2\|_F^2} (\alpha_1,\ \alpha_2) \in T := \mathbb{R}^{2\times w\times w}$$

where $\|\cdot\|_F$ is the Frobenius norm.

Remark C.3. The matrices α_1 and α_2 can be replaced by convolutions $\tilde{\alpha}_j = \varphi * \alpha_j$ with a smooth kernel φ in order to obtain a smooth vector field τ.

Define the norm[3] of $\tau \in T = \mathbb{R}^{2\times w\times w}$ by

$$\|\tau\|_T = \max_{p,q=1,\dots,w} \sqrt{\tau_{1,p,q}^2 + \tau_{2,p,q}^2}$$

and let

$$k^n := \arg\min_{k\neq l} \|\tau(\mathbf{x}^n, k)\|_T$$

$$\tau^n := \tau(\mathbf{x}^n, k^n).$$

Finally, we define the deformed image $\mathbf{x}^{n+1} \in X$ by

$$\mathbf{x}_{i,p,q}^{n+1} = \xi_i \left(\frac{p}{w+1} + \tau_{1,p,q}^n,\ \frac{q}{w+1} + \tau_{2,p,q}^n \right)$$

for $p, q = 1, \dots, w$ and $i = 1, 2, 3$. The values of ξ can be estimated by interpolation from $\mathbf{x}^n$.

If $l \neq \arg\max_k F_k(\mathbf{x}^{n+1})$, then return $\hat{\mathbf{x}} := \mathbf{x}^{n+1}$. Otherwise, repeat the procedure above for $n+1$.

[3] We choose this norm to be consistent with the theory in section C.1.2. One might want to minimize the Frobenius norm of τ instead.