Minimally distorted Adversarial Examples with a Fast Adaptive Boundary Attack

Francesco Croce [1] **Matthias Hein** [1]

Abstract

The evaluation of robustness against adversarial manipulation of neural networks-based classifiers is mainly tested with empirical attacks as methods for the exact computation, even when available, do not scale to large networks. We propose in this paper a new white-box adversarial attack wrt the l_p-norms for $p \in \{1, 2, \infty\}$ aiming at finding the minimal perturbation necessary to change the class of a given input. It has an intuitive geometric meaning, yields quickly high quality results, minimizes the size of the perturbation (so that it returns the robust accuracy at every threshold with a single run). It performs better or similar to state-of-the-art attacks which are partially specialized to one l_p-norm, and is robust to the phenomenon of gradient masking.

1. Introduction

The finding of the vulnerability of neural networks-based classifiers to adversarial examples, that is small perturbations of the input able to modify the decision of the models, started a fast development of a variety of attack algorithms. The high effectiveness of adversarial attacks reveals the fragility of these networks which questions their safe and reliable use in the real world, especially in safety critical applications. Many defenses have been proposed to fix this issue (Gu & Rigazio, 2015; Zheng et al., 2016; Papernot et al., 2016; Huang et al., 2016; Bastani et al., 2016; Madry et al., 2018), but with limited success, as new more powerful attacks showed (Carlini & Wagner, 2017b; Athalye et al., 2018; Mosbach et al., 2018). In order to trust the decision of a model, it is necessary to evaluate the exact adversarial robustness. Although this is possible for ReLU networks (Katz et al., 2017; Tjeng et al., 2019) these techniques do not scale to commonly used large networks. Thus, the robustness is evaluated approximating the solution of the minimal adversarial perturbation problem through adversarial attacks.

One can distinguish attacks into black-box (Narodytska & Kasiviswanathan, 2016; Brendel et al., 2018; Su et al., 2019), where one is only allowed to query the classifier, and white-box attacks, where one has full control over the network, according to the attack model used to create adversarial examples (typically some l_p-norm, but others have become popular as well, e.g. (Brown et al., 2017; Engstrom et al., 2017; Wong et al., 2019)), whether they aim at the minimal adversarial perturbation (Carlini & Wagner, 2017a; Chen et al., 2018; Croce et al., 2019) or rather any perturbation below a threshold (Kurakin et al., 2017; Madry et al., 2018; Zheng et al., 2019), if they have lower (Moosavi-Dezfooli et al., 2016; Modas et al., 2019) or higher (Carlini & Wagner, 2017a; Croce et al., 2019) computational cost. Moreover, it is clear that due to the non-convexity of the problem there exists no universally best attack (apart from the exact methods), since this depends on runtime constraints, networks architecture, dataset, etc. However, our goal is to have an attack which performs well under a broad spectrum of conditions with minimal amount of hyperparameter tuning.

In this paper we propose a new white-box attack scheme which performs comparably or better than established attacks and has the following features: first, it aims at adversarial samples with *minimal distance* to the attacked point, measured wrt the l_p-norms with $p \in \{1, 2, \infty\}$. Compared to the popular PGD (projected gradient descent)-attack of (Madry et al., 2018) this has the clear advantage that our method does not need to be restarted for every threshold ϵ if one wants to evaluate the success rate of the attack with perturbations constrained to be in $\{\delta \in \mathbb{R}^d \mid \|\delta\|_p \leq \epsilon\}$. Thus it is particularly suitable to get a complete picture on the robustness of a classifier with low computational cost. Second, it achieves *fast* good quality in terms of average distortion or robust accuracy. At the same time we show that increasing the number of restarts keeps improving the results and makes it competitive to or stronger than the strongest available attacks. Third, although it comes with a few parameters, these generalize well across datasets, architectures and norms considered, so that we have an almost *off-the-shelf method*. Most importantly, unlike PGD and

[1] University of Tübingen, Germany. Correspondence to: F. Croce <francesco.croce@uni-tuebingen.de>.

Proceedings of the 37^{th} International Conference on Machine Learning, Online, PMLR 119, 2020. Copyright 2020 by the author(s).

other methods, there is no step size parameter, which potentially has to be carefully adapted to every new network, and we show that it is scaling invariant. Both properties lead to the fact that it is robust to gradient masking which can be a problem for PGD (Tramèr & Boneh, 2019).

2. FAB: a Fast Adaptive Boundary Attack

We first introduce minimal adversarial perturbations, then we recall the definition and properties of the projection wrt the l_p-norms of a point on the intersection of a hyperplane and box constraints, as they are an essential part of our attack. Finally, we present our FAB-attack algorithm to generate minimally distorted adversarial examples.

2.1. Minimal adversarial examples

Let $f : \mathbb{R}^d \to \mathbb{R}^K$ be a classifier which assigns every input $x \in \mathbb{R}^d$ (with d the dimension of the input space) to one of the K classes according to $\underset{r=1,\ldots,K}{\arg\max} f_r(x)$. In many scenarios the input of f has to satisfy a specific set of constraints C, e.g. images are represented as elements of $[0, 1]^d$. Then, given a point $x \in \mathbb{R}^d$ with true class c, we define the *minimal adversarial perturbation* for x wrt the l_p-norm as

$$\begin{aligned} \delta_{\text{min},p} = &\underset{\delta\in\mathbb{R}^d}{\arg\min}\ \left\|\delta\right\|_p, \\ \text{s.th.}\quad &\max_{l\neq c}\ f_l(x+\delta) \geq f_c(x+\delta), \quad x+\delta \in C. \end{aligned} \tag{1}$$

The optimization problem (1) is non-convex and NP-hard for non-trivial classifiers (Katz et al., 2017) and, although for some classes of networks it can be formulated as a mixed-integer program (Tjeng et al., 2019), the computational cost of solving it is prohibitive for large, normally trained networks. Thus, $\delta_{\text{min},p}$ is usually approximated by an *attack algorithm*, which can be seen as a heuristic to solve (1). We will see in the experiments that current attacks sometimes drastically overestimate $\left\|\delta_{\text{min},p}\right\|_p$ and thus the robustness of the networks.

2.2. Projection on a hyperplane with box constraints

Let $w \in \mathbb{R}^d$ and $b \in \mathbb{R}$ be the normal vector and the offset defining the hyperplane $\pi : \langle w, x\rangle + b = 0$. Let $x \in \mathbb{R}^d$, we denote by the *box-constrained projection* wrt the l_p-norm of x on π (projection onto the intersection of the box $C = \{z \in \mathbb{R}^d : l_i \leq z_i \leq u_i\}$ and the hyperplane π) the following minimization problem:

$$\begin{aligned} z^* = &\underset{z\in\mathbb{R}^d}{\arg\min}\ \left\|z-x\right\|_p \\ \text{s.th.}\quad &\langle w, z\rangle + b = 0, \quad l_i \leq z_i \leq u_i,\ i = 1,\ldots,d, \end{aligned} \tag{2}$$

where $l_i, u_i \in \mathbb{R}$ are lower and upper bounds on each component of z. For $p \geq 1$ the optimization problem (2) is convex. (Hein & Andriushchenko, 2017) proved that for $p \in \{1, 2, \infty\}$ the solution can be obtained in $\mathcal{O}(d \log d)$ time, that is the complexity of sorting a vector of d elements, as well as determining that there exists no feasible point.

Since this projection is part of our iterative scheme, we need to handle specifically the case of (2) being infeasible. In this case, defining $\rho = \text{sign}(\langle w, x\rangle + b)$, we instead compute $z' = \underset{l_i\leq z_i\leq u_i}{\arg\min}\, \rho\cdot(\langle w, z\rangle + b)$, whose solution is

$$z'_i = \begin{cases} l_i & \text{if } \rho w_i > 0, \\ u_i & \text{if } \rho w_i < 0, \ \text{ for } i = 1,\ldots,d. \\ x_i & \text{if } w_i = 0 \end{cases} \tag{3}$$

Assuming that the point x satisfies the box constraints (as it holds in our algorithm), this is equivalent to identifying the corner of the d-dimensional box, defined by the componentwise constraints on z, closest to the hyperplane π. Note that if (2) is infeasible then the objective function of (3) stays positive and the points x and z are strictly contained in the same of the two halfspaces divided by π. Finally, we define the projection operator

$$\text{proj}_p : (x, \pi, C) \longmapsto \begin{cases} z^* & \text{if (2) is feasible} \\ z' & \text{else} \end{cases} \tag{4}$$

which yields the point as close as possible to π without violating the box constraints.

2.3. FAB-attack

We introduce now our algorithm to produce minimally distorted adversarial examples, wrt any l_p-norm for $p \in \{1, 2, \infty\}$, for a given point x_{orig} initially correctly classified by f as class c. The high-level idea is that, first, we use the linearization of the classifier at the current iterate $x^{(i)}$ to compute the box-constrained projections of $x^{(i)}$ respectively x_{orig} onto the approximated decision hyperplane and, second, we take a convex combinations of these projections depending on the distance of $x^{(i)}$ and x_{orig} to the decision hyperplane. Finally, we perform an extrapolation step. We explain below the geometric motivation behind these steps. The attack closest in spirit is DeepFool (Moosavi-Dezfooli et al., 2016) which is known to be very fast but suffers from low quality. DeepFool just tries to find the decision boundary quickly but has no incentive to provide a solution close to x_{orig}. Our scheme resolves this main problem and, together with the exact projection we use, leads to a principled way to track the decision boundary (the surface where the decision of f changes) *close* to x_{orig}.

If f was a linear classifier then the closest point to $x^{(i)}$ on the decision hyperplane could be found in closed form. However neural networks are highly non-linear (although ReLU networks, i.e. neural networks which use ReLU as activation function, are piecewise affine functions and thus locally a linearization of the network is an exact description of the classifier). Let $l \neq c$, then the decision boundary between classes l and c can be locally approximated using a first order Taylor expansion at $x^{(i)}$ by the hyperplane

$$\begin{aligned}\pi_l(z) : \; & f_l(x^{(i)}) - f_c(x^{(i)}) \\ & + \left\langle \nabla f_l(x^{(i)}) - \nabla f_c(x^{(i)}), z - x^{(i)} \right\rangle = 0.\end{aligned} \tag{5}$$

Moreover the l_p-distance $d_p(x^{(i)}, \pi_l)$ of $x^{(i)}$ to π_l is given, assuming $\frac{1}{p} + \frac{1}{q} = 1$, by

$$d_p(x^{(i)}, \pi_l) = \frac{|f_l(x^{(i)}) - f_c(x^{(i)})|}{\nabla f_l(x^{(i)}) - \nabla f_c(x^{(i)}) \quad {}_q}. \tag{6}$$

Note that if $d_p(x^{(i)}, \pi_l) = 0$ then $x^{(i)}$ belongs to the true decision boundary. Moreover, if the local linear approximation of the network is correct then the class s with the decision hyperplane closest to the point $x^{(i)}$ can be computed as

$$s = \operatorname*{arg\,min}_{l \neq c} \frac{|f_l(x^{(i)}) - f_c(x^{(i)})|}{\nabla f_l(x^{(i)}) - \nabla f_c(x^{(i)}) \quad {}_q}. \tag{7}$$

Thus, given that the approximation holds in some large enough neighborhood, the projection $\mathrm{proj}_p(x^{(i)}, \pi_s, C)$ of $x^{(i)}$ onto π_s lies on the decision boundary (unless (2) is infeasible).

Biased gradient step: The iterative algorithm $x^{(i+1)} = \mathrm{proj}_p(x^{(i)}, \pi_s, C)$ would be similar to DeepFool except that our projection operator is exact whereas they project onto the hyperplane and then clip to $[0, 1]^d$. This scheme is not biased towards the original target point x_{orig}, thus it goes typically further than necessary to find a point on the decision boundary as basically the algorithm does not aim at the minimal adversarial perturbation. Then we consider additionally $\mathrm{proj}_p(x_{\text{orig}}, \pi_s, C)$ and use instead the iterative step, with $x^{(0)} = x_{\text{orig}}$ and $\alpha \in [0, 1]$, defined as

$$x^{(i+1)} = (1 - \alpha)\, \mathrm{proj}_p(x^{(i)}, \pi_s, C) + \alpha\, \mathrm{proj}_p(x_{\text{orig}}, \pi_s, C), \tag{8}$$

which biases the step towards x_{orig} (see Figure 1). Note that this is a convex combination of two points on π_s and in C and thus also $x^{(i+1)}$ lies on π_s and is contained in C.

As we wish a scheme with minimal amount of parameters, our goal is an automatic selection of α based on the available geometric quantities. Let

$$\begin{aligned}\delta^{(i)} &= \mathrm{proj}_p(x^{(i)}, \pi_s, C) - x^{(i)}, \\ \delta^{(i)}_{\text{orig}} &= \mathrm{proj}_p(x_{\text{orig}}, \pi_s, C) - x_{\text{orig}}.\end{aligned}$$

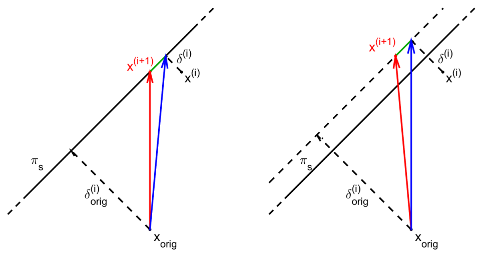

Figure 1. Visualization of FAB-attack scheme: Left, case $\eta = 1$, right, $\eta > 1$ (extrapolation). In blue we show $\mathrm{proj}_p(x^{(i)}, \pi_s, C)$, the iterate one would get without any bias towards x_{orig}, in green the effect of the bias we introduce and in red the actual iterate $x^{(i+1)}$ of FAB-attack in (10). FAB-attack stays closer to x_{orig} compared to the unbiased gradient step with $\mathrm{proj}_p(x^{(i)}, \pi_s, C)$.

Note that $\delta^{(i)} \; {}_p$ and $\delta^{(i)}_{\text{orig}} \; {}_p$ are the distances of $x^{(i)}$ and x_{orig} to π_s (inside C). We propose to use for the parameter α the relative magnitude of these two distances, that is

$$\alpha = \min \left\{ \frac{\delta^{(i)} \; {}_p}{\delta^{(i)} \; {}_p + \; \delta^{(i)}_{\text{orig}} \; {}_p}, \alpha_{\text{max}} \right\} \in [0, 1]. \tag{9}$$

The motivation for doing so is that if $x^{(i)}$ is close to the decision boundary then we should stay close to this point (note that π_s is the approximation of f computed at $x^{(i)}$ and thus it is valid in a small neighborhood of $x^{(i)}$, whereas x_{orig} is farther away). On the other hand we want to have the bias towards x_{orig} in order not to go too far away from x_{orig}. This is why α depends on the distances of $x^{(i)}$ and x_{orig} to π_s but we limit it from above with α_{max}. Finally, we use a small extrapolation step as we noted empirically, similarly to (Moosavi-Dezfooli et al., 2016), that this helps to cross faster the decision boundary and get an adversarial sample. This leads to the final scheme:

$$x^{(i+1)} = \mathrm{proj}_C\Big((1 - \alpha)\big(x^{(i)} + \eta\delta^{(i)}\big) + \alpha\big(x_{\text{orig}} + \eta\delta^{(i)}_{\text{orig}}\big)\Big), \tag{10}$$

where α is chosen as in (9), $\eta \geq 1$ and proj_C is the projection onto the box which can be done by clipping. In Figure 1 we visualize the scheme: in black one can see the hyperplane π_s and the vectors $\delta^{(i)}_{\text{orig}}$ and $\delta^{(i)}$, in blue the step not biased towards x_{orig}, while in red the biased step of FAB-attack, see (10). The green vector shows the bias towards the original point we introduce. On the left of Figure 1 we use $\eta = 1$, while on the right we use extrapolation $\eta > 1$.

Interpretation of $\mathrm{proj}_p(x_{\text{orig}}, \pi_s, C)$: The projection of the target point x_{orig} onto the intersection of π_s and C is

$$\operatorname*{arg\,min}_{z \in \mathbb{R}^d} \; \|z - x_{\text{orig}}\|_p \;\; \text{s.th.} \;\; \langle w, z \rangle + b = 0, \;\; l_i \leq z_i \leq u_i,$$

Note that replacing z by $x^{(i)} + \delta$ we can rewrite this as

$$\begin{aligned}&\mathop{\arg\min}_{\delta\in\mathbb{R}^d} \quad x^{(i)} + \delta - x_{\text{orig}} \;\;_p\\ &\text{s.th.} \quad \left\langle w, x^{(i)} + \delta \right\rangle + b = 0, \quad l_i \leq x_i + \delta_i \leq u_i.\end{aligned}$$

This can be interpreted as the minimization of the distance of the next iterate $x^{(i)} + \delta$ to the target point x_{orig} so that $x^{(i)} + \delta$ lies on the intersection of the (approximate) decision hyperplane and the box C. This point of view on the projection $\text{proj}_p(x_{\text{orig}}, \pi_s, C)$ again justifies using a convex combination of the two projections in our scheme in (10).

Backward step: The described scheme finds in a few iterations adversarial perturbations. However, we are interested in minimizing their norms. Thus, once we have a new point $x^{(i+1)}$, we check whether it is assigned by f to a class different from c. In this case, we apply

$$x^{(i+1)} = (1 - \beta)x_{\text{orig}} + \beta x^{(i+1)}, \quad \beta \in (0, 1), \qquad (11)$$

that is we go back towards x_{orig} on the segment $[x^{(i+1)}, x_{\text{orig}}]$, effectively starting again the algorithm at a point which is close to the decision boundary. In this way, due to the bias of the method towards x_{orig} we successively find adversarial perturbations of smaller norm, meaning that the algorithm *tracks* the decision boundary while getting closer to x_{orig}. We fix $\beta = 0.9$ in all experiments.

Final search: Our scheme finds points close to the decision boundary but often they are slightly off as the linear approximation is not exact and we apply the extrapolation step with $\eta > 1$. Thus, after finishing N_{iter} iterations of our algorithmic scheme, we perform a last, fast step to further improve the quality of the adversarial examples. Let x_{out} be the closest point to x_{orig} classified differently from c, say $s \neq c$, found with the iterative scheme. It holds that $f_s(x_{\text{out}}) - f_c(x_{\text{out}}) > 0$ and $f_s(x_{\text{orig}}) - f_c(x_{\text{orig}}) < 0$. This means that, assuming f continuous, there exists a point x^* on the segment $[x_{\text{out}}, x_{\text{orig}}]$ such that $f_s(x^*) - f_c(x^*) = 0$ and $\|x^* - x_{\text{orig}}\|_p < \|x_{\text{out}} - x_{\text{orig}}\|_p$. If f is linear

$$x^* = x_{\text{out}} - \frac{(f_s(x_{\text{out}}) - f_c(x_{\text{out}}))\,(x_{\text{out}} - x_{\text{orig}})}{f_s(x_{\text{out}}) - f_c(x_{\text{out}}) + f_s(x_{\text{orig}}) - f_c(x_{\text{orig}})}. \qquad (12)$$

Since f is non-linear, we compute iteratively for a few steps

$$x_{\text{temp}} = x_{\text{out}} - \frac{(f_s(x_{\text{out}}) - f_c(x_{\text{out}}))\,(x_{\text{out}} - x_{\text{orig}})}{f_s(x_{\text{out}}) - f_c(x_{\text{out}}) + f_s(x_{\text{orig}}) - f_c(x_{\text{orig}})}, \qquad (13)$$

each time replacing in (13) x_{out} with x_{temp} if $f_s(x_{\text{temp}}) - f_c(x_{\text{temp}}) > 0$ or x_{orig} with x_{temp} if instead $f_s(x_{\text{temp}}) - f_c(x_{\text{temp}}) < 0$. With this kind of modified binary search one can find a better adversarial sample with the cost of a few forward passes (which is fixed to 3 in all experiments).

Algorithm 1 FAB-attack

```
Input : $x_{\text{orig}}$ original point, $c$ original class,
        $N_{\text{restarts}}, N_{\text{iter}}, \alpha_{\max}, \beta, \eta, \epsilon, p$
Output : $x_{\text{out}}$ adversarial example
$u \leftarrow +\infty$
for $j = 1, \ldots, N_{restarts}$ do
    if $j = 1$ then $x^{(0)} \leftarrow x_{\text{orig}}$;
    else $x^{(0)} \leftarrow$ randomly sampled s.th. $x^{(0)} - x_{\text{orig}}\;_p = \min\{u,\epsilon\}/2$;
    for $i = 0, \ldots, N_{iter} - 1$ do
        $s \leftarrow \mathop{\arg\min}_{l\neq c} \frac{|f_l(x^{(i)}) - f_c(x^{(i)})|}{\|\nabla f_l(x^{(i)}) - \nabla f_c(x^{(i)})\|_q}$
        $\delta^{(i)} \leftarrow \text{proj}_p(x^{(i)}, \pi_s, C)$
        $\delta^{(i)}_{\text{orig}} \leftarrow \text{proj}_p(x_{\text{orig}}, \pi_s, C)$
        compute $\alpha$ as in Equation (9)
        $x^{(i+1)} \leftarrow \text{proj}_C\Big((1-\alpha)\left(x^{(i)} + \eta\delta^{(i)}\right)$
                        $+ \alpha(x_{\text{orig}} + \eta\delta^{(i)}_{\text{orig}})\Big)$
        if $x^{(i+1)}$ is not classified as $c$ then
            if $x^{(i+1)} - x_{orig}\;_p < u$ then
                $x_{\text{out}} \leftarrow x^{(i+1)}$
                $u \leftarrow \; x^{(i+1)} - x_{\text{orig}}\;_p$
            end
            $x^{(i+1)} \leftarrow (1-\beta)x_{\text{orig}} + \beta x^{(i+1)}$
        end
    end
end
perform 3 steps of final search on $x_{\text{out}}$ as in (13)
```

Random restarts: So far all the steps are deterministic. To improve the results, we introduce the option of random restarts, that is $x^{(0)}$ is randomly sampled in the proximity of x_{orig} instead of being x_{orig} itself. Most attacks benefit from random restarts, e.g. (Madry et al., 2018; Zheng et al., 2019), especially dealing with models trained for robustness (Mosbach et al., 2018), as it allows a wider exploration of the input space. We choose to sample from the l_p-sphere centered in the original point with radius half the l_p-norm of the current best adversarial perturbation (or a given threshold if no adversarial example has been found yet).

Computational cost: Our attack, in Algorithm 1, consists of two main operations: the computation of f and its gradients and solving the projection (2). We perform, for each iteration, a forward and a backward pass of the network in the gradient step and a forward pass in the backward step. The projection can be efficiently implemented to run in batches on the GPU and its complexity depends only on the input dimension. Thus, except for shallow models, its cost is much smaller than the passes through the network. We can approximate the computational cost of

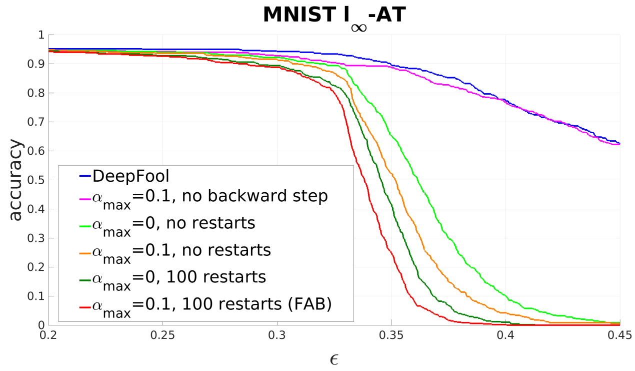

Figure 2. **Ablation study to DeepFool for l_∞-attacks.** The curves show the robust accuracy as a function of the threshold ϵ under different attacks on the l_∞-AT model on MNIST (lower values mean stronger attacks). The introduction of the convex combination ($\alpha_{\max} = 0.1$, no backward step) already improves over DeepFool. If one uses the backward step, the case $\alpha_{\max} = 0$ (which can be seen as an improved iterative DF) is worse than $\alpha_{\max} = 0.1$ with the same number of restarts.

our algorithm by the total number of calls of the classifier $N_{\text{iter}} \times N_{\text{restarts}} \times (2 \times \text{forward passes} + 1 \times \text{backward pass})$ Per restart one has to add the three forward passes for the final search.

2.4. Scale Invariance of FAB-attack

For a given classifier f, the decisions and thus adversarial samples do not change if we rescale the classifier $g = \alpha f$ for $\alpha > 0$ or shift its logits as $h = f + \beta$ for $\beta \in \mathbb{R}$. The following proposition states that FAB-attack is invariant under both rescaling and shifting (proof in supplement).

Proposition 2.1 *Let $f : \mathbb{R}^d \to \mathbb{R}^K$ be a classifier. Then for any $\alpha > 0$ and $\beta \in \mathbb{R}$ the output x_{out} of Algorithm 1 for the classifier f is the same as of the classifiers $g = \alpha f$ and $h = f + \beta$.*

We note that the cross-entropy loss $\text{CE}(x, y, f) = -\log(e^{f_y(x)} / \sum_{j=1}^K e^{f_j(x)})$ used as objective in the normal PGD attack and its gradient wrt x

$$\nabla_x \text{CE}(x, y, f) = -\nabla_x f_y(x) + \frac{\sum_{j=1}^K e^{f_j(x)} \nabla_x f_j(x)}{\sum_{j=1}^K e^{f_j(x)}}$$

are not invariant under rescaling. Moreover, we observe that the gradient vanishes for αf if $f_y(x) > f_j(x)$ for $j \neq y$ (correctly classified point) as $\alpha \to \infty$. Due to finite precision the gradient becomes zero for finite α and it is obvious that in this case PGD gets stuck. Due to the rescaling invariance FAB-attack is not affected by gradient masking due to this phenomenon as it uses the gradient of the differences of the logits and not the gradient of the cross-entropy loss. The latter one runs much earlier into numerical problems when one upscales the classifier due to the exponential function. In the experiments (see below) we show that PGD can catastrophically fail due to a "wrong" scaling whereas FAB-attack is unaffected.

2.5. Comparison to DeepFool

The idea of exploiting the first order local approximation of the decision boundary is not novel but the basis of one of the first white-box adversarial attacks, DeepFool (DF) from (Moosavi-Dezfooli et al., 2016). While DF and our FAB-attack share the strategy of using a linear approximation of the classifier and projecting on the decision hyperplanes, we want to point out many key differences: first, DF does not solve the projection (2) but its simpler version without box constraints, clipping afterwards. Second, their gradient step does not have any bias towards the original point, that is equivalent to $\alpha = 0$ in (10). Third, DF does not have any backward step, final search or restart, as it stops as soon as a misclassified point is found (its goal is to provide quickly an adversarial perturbation of average quality).

We perform an ablation study of the differences to DF in Figure 2, where we show robust accuracy as a function of the threshold ϵ (lower is better). We present the results of DeepFool (blue) and FAB-attack with the following variations: $\alpha_{\max} = 0.1$ and no backward step (magenta), $\alpha_{\max} = 0$ (that is no bias in the gradient step) and no restarts (light green), $\alpha_{\max} = 0.1$ and no restarts (orange), $\alpha_{\max} = 0$ and 100 restarts (dark green) and $\alpha_{\max} = 0.1$ and 100 restarts, that is FAB-attack, (red). We can see how every addition we make to the original scheme of DeepFool contributes to the significantly improved performance of FAB-attack when compared to the original DeepFool.

3. Experiments

Models: We run experiments on MNIST, CIFAR-10 (Krizhevsky et al., 2014) and Restricted ImageNet (Tsipras et al., 2019). For each dataset we consider a normally trained model (*plain*) and two adversarially trained ones as in (Madry et al., 2018) wrt the l_∞-norm (l_∞-AT) and the l_2-norm (l_2-AT) (see supplementary material for details).

Attacks: We compare the performance of FAB-attack[1] to those of attacks representing the state-of-the-art in each norm: DeepFool (DF) (Moosavi-Dezfooli et al., 2016), Carlini-Wagner l_2-attack (CW) (Carlini & Wagner, 2017a), Linear Region l_2-Attack (LRA) (Croce et al., 2019), Projected Gradient Descent on the cross-entropy function (PGD) (Kurakin et al., 2017; Madry et al., 2018; Tramèr & Boneh, 2019), Distributionally Adversarial Attack (DAA) (Zheng et al., 2019), SparseFool (SF) (Modas et al., 2019),

[1] https://github.com/fra31/fab-attack

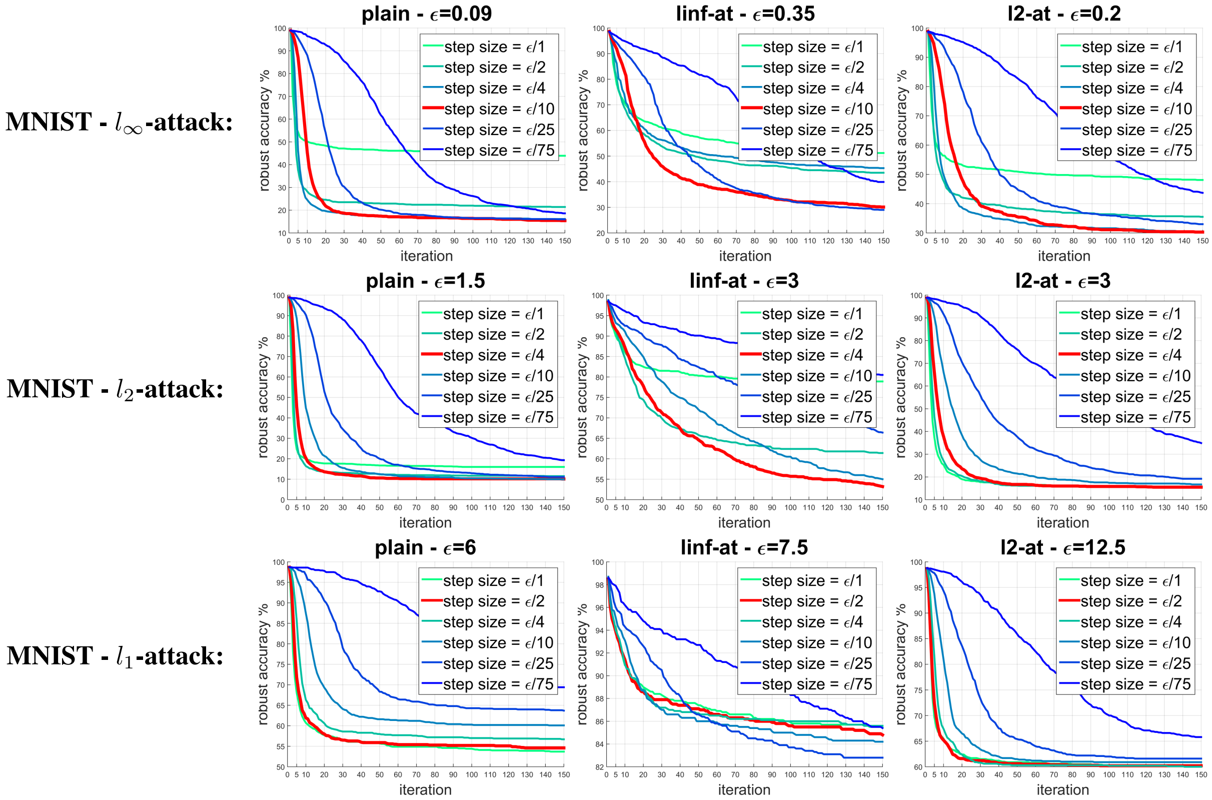

Figure 3. Evolution of robust accuracy across iterations for different step sizes for PGD wrt l_1, l_2, l_∞ for three different models on MNIST. In red we highlight the step size we used for each norm in the experiments. Notice that it performs on average the best. Here we evaluate on MNIST - the supplementary material contains also CIFAR-10 and other thresholds.

Elastic-net Attack (EAD) (Chen et al., 2018). We use DF from (Rauber et al., 2017), CW and EAD as in (Papernot et al., 2017), DAA and LRA with the code from the original papers, while we reimplemented SF and PGD. For MNIST and CIFAR-10 we used DAA with 50 restarts, PGD and FAB with 100 restarts. For Restricted ImageNet, we used DAA, PGD and FAB with 10 restarts (for l_1 we used 5 restarts, since the methods benefit from more iterations). Moreover, we could not use LRA since it hardly scales to models of such scale and CW and EAD for compatibility issues between the implementations of attacks and models. See the supplementary material for more details e.g. regarding number of iterations and hyperparameters of all attacks. In particular, we provide a detailed analysis of the dependency of PGD on the step size. Indeed the optimal choice of the step size is quite important for PGD. In order to select the optimal step size for PGD for each norm, we performed a grid search on the step size parameter in ϵ/t for $t \in \{1, 2, 4, 10, 25, 75\}$ for different models and thresholds, and took the values working best on average, see Figure 3 for an illustration (similar plots for other datasets and thresholds are presented in the supplements). As a result we use for PGD wrt l_∞ step size $\epsilon/10$ and the direction is the sign of the gradient of the cross entropy loss, for PGD wrt l_2 we do a step in the direction of the l_2-normalized gradient with step size $\epsilon/4$, for PGD wrt l_1 we use the gradient step suggested in (Tramèr & Boneh, 2019) (with sparsity levels of 1% for MNIST and 10% for CIFAR-10 and Restricted ImageNet) with step size $\epsilon/2$. For FAB-attack we use always $\beta = 0.9$ and on MNIST and CIFAR-10: $\alpha_{\max} = 0.1$, $\eta = 1.05$ and on Restricted ImageNet: $\alpha_{\max} = 0.05$, $\eta = 1.3$. These parameters are the same for all norms.

Evaluation metrics: The *robust accuracy* for a threshold ϵ is the classification accuracy (in percentage) when an adversary is allowed to change every test input with perturbations of l_p-norm smaller than ϵ in order to change the decision. Thus stronger attacks produce lower robust accuracies. For each model and dataset we fix five thresholds at which we compute the robust accuracy for each attack (we choose the thresholds so that the robust accuracy covers the range between clean accuracy and 0). We evaluate the attacks by the following statistics: i) **avg. rob. accuracy**: the mean of the robust accuracies achieved by the attack over all models and thresholds (lower is better), ii) **# best**: how many times the attack achieves the lowest robust accuracy (it is the most effective), iii) **avg. difference to best**: for each model/threshold we compute the difference between

Table 1. Performance summary of all attacks on MNIST and CIFAR-10 (aggregated). We report, for each norm, "avg. rob. acc.", the mean of robust accuracies across all models and datasets (lower is better), "# best", number of times the attack is the best one, "avg. diff. to best" and "max diff. to best", the mean and maximum difference of the robust accuracy of the attack to the robust accuracy of the best attack for each model/threshold (on the first 1000 points for l_∞ and l_1, 500 for l_2, of the test sets). The numbers after the name of the attacks indicate the number of restarts. In total we have 5 thresholds × 6 models = 30 cases for each of the 3 norms. *Note that for FAB-10 (i.e. with 10 restarts) the "# best" is computed excluding the results of FAB-100.

statistics on MNIST + CIFAR-10

l_∞-norm		DF	DAA-50	PGD-100	FAB-10	FAB-100
avg. rob. acc.		58.81	60.67	46.07	46.18	**45.47**
# best		0	8	12	13*	**17**
avg. diff. to best		14.58	16.45	1.85	1.96	**1.25**
max diff. to best		78.10	49.00	**10.70**	20.30	17.10
l_2-norm	CW	DF	LRA	PGD-100	FAB-10	FAB-100
avg. rob. acc.	45.09	56.10	36.97	44.94	36.41	**35.57**
# best	4	1	9	11	19*	**23**
avg. diff. to best	9.65	20.67	1.54	9.51	0.98	**0.13**
max diff. to best	65.40	91.40	13.60	64.80	8.40	**1.60**
l_1-norm		SF	EAD	PGD-100	FAB-10	FAB-100
avg. rob. acc.		64.47	35.79	49.51	33.26	**29.46**
# best		0	13	0	10*	**17**
avg. diff. to best		35.31	6.63	20.35	4.10	**0.30**
max diff. to best		95.90	58.40	74.00	21.80	**1.60**

Table 2. As in Table 1 statistics of the performance of different attacks on Restricted ImageNet (on the first 500 points of the validation set). In total we consider 5 thresholds × 3 models = 15 cases for each of the 3 norms.

statistics on Restricted ImageNet

	l_∞-norm				**l_2-norm**			**l_1-norm**		
	DF	DAA-10	PGD-10	FAB-10	DF	PGD-10	FAB-10	SF	PGD-5	FAB-5
avg. rob. acc.	35.61	38.44	**26.91**	27.83	45.69	**31.75**	33.24	71.31	40.64	**38.12**
# best	0	1	**13**	3	0	**14**	1	0	3	**12**
avg. diff. best	8.75	11.57	**0.04**	0.96	13.99	**0.04**	1.53	33.52	2.85	**0.33**
max diff. best	14.60	37.20	**0.40**	2.00	25.40	**0.60**	3.40	59.00	6.20	**2.40**

the robust accuracy of the attack and the best one across all the attacks, then we average over all models/thresholds, iv) **max difference to best**: as "avg. difference to best", but with the maximum difference instead of the average one. In the supplement we report additionally the average l_p-norm of the adversarial perturbations given by the attacks.

Results: We report the complete results in the supplementary material, while we summarize them in Table 1 (MNIST and CIFAR-10 aggregated, as we used the same attacks) and Table 2 (Restricted ImageNet). Our FAB-attack achieves the best results in all statistics for every norm (with the only exception of "max diff. to best" in l_∞) on MNIST+CIFAR-10. In particular, while on l_∞ the "avg. robust accuracy" of PGD is not far from that of FAB, the gap is large when considering l_2 and l_1 (in the appendix we provide the aggregate statistics without the result on l_∞-AT MNIST, showing that FAB is still the best attack even if one leaves out this failure case of PGD). Interestingly, the second best attack in terms of average robust accuracy, is different for every norm (PGD for l_∞, LRA for l_2, EAD for l_1), which implies that FAB outperforms algorithms specialized in the individual norms.

We also report the results of FAB-10, that is our attack with only 10 restarts, to show that FAB yields high quality results already with a low budget in terms of time/computational cost. In fact, FAB-10 has "avg. robust accuracy" better than or very close to that of the strongest versions of the other attacks (see below for a runtime analysis, where one observes that FAB-10 is the fastest attack when excluding the significantly worse DF and SF attacks). On Restricted ImageNet, FAB-attack gets the best results in all statistics for l_1, while for l_∞ and l_2 PGD performs on average better, but the difference in "avg. robust accuracy" is small.

In general, both average and maximum *difference to best* of FAB-attack are small for all the datasets and norms, implying that it does not suffer from severe failures, which makes it an efficient, high quality technique to evaluate the robustness of classifiers for all l_p-norms. Finally, we show in the supplementary material that FAB-attack outperforms or matches the competitors in 16 out of 18 cases when comparing the average l_p-norms of the generated adversarial perturbations.

Table 3. We attack the ResNet-110 in (Pang et al., 2020) on CIFAR-10 at $\epsilon = 8/255$. The performance of the PGD attack on the cross entropy loss (CE) heavily depends on both scale of the classifier and the step size. In contrast, the scaling invariant FAB-attack works well even on the original (unscaled) model.

attack	*step size*	*robust accuracy*
PGD-CE	$\epsilon/10$	90.2%
PGD-CE	$\epsilon/2$	90.2%
PGD-CE rescaled	$\epsilon/10$	12.9%
PGD-CE rescaled	$\epsilon/2$	2.5%
FAB	-	**0.3%**

Resistance to gradient masking: It has been argued (Tramèr & Boneh, 2019) that models trained with first-order methods to be resistant wrt l_∞-attacks on MNIST (adv. training) give a false sense of robustness in l_1 and l_2 due to *gradient masking*. This means that standard gradient-based methods like PGD have problems to find adversarial examples while they still exist. In contrast, FAB does not suffer from gradient masking. In Table 8 (supplement) we see that it is extremely effective also wrt l_1 and l_2 on the l_∞-robust model, outperforming by a large margin the competitors. The reason is that FAB is not dependent on the norm of the gradient but just its direction matters for the definition of the hyperplane in (5). While we believe that resistance to gradient masking is a key property of a solid attack, we recompute the statistics of Table 1 excluding l_1 and l_2 attacks on the l_∞-AT model on MNIST (see supplements). FAB still achieves in most of the cases the best aggregated statistics, implying that our attack is effective whether or not the attacked classifier tends to "mask" the gradient.

We have shown in Section 2.4 that FAB-attack is invariant under rescaling of the classifier. We provide an example why this is a desirable property of an adversarial attack. We consider the defense proposed in (Pang et al., 2020), in particular their ResNet-110 (without adversarial training) for CIFAR-10. In Table 5 in (Pang et al., 2020) it is claimed that this model has a robust accuracy of 31.4% for $8/255$ obtained by a PGD attack on their new loss function. They say that a standard PGD attack on the cross-entropy loss performs much worse. We test the performance of PGD on the cross-entropy loss, both using the original classifier and the same scaled down by a factor of 10^6. Moreover, we use the default step size $\epsilon/10$ together with $\epsilon/2$. The results are reported in Table 3. We can see that PGD on the original model yields more than 90% robust accuracy which confirms the statement in (Pang et al., 2020) about the cross-entropy loss being unsuitable for this case. However, PGD applied to the rescaled classifier reduces robust accuracy below 13%. The better step size $\epsilon/2$ decreases it to 2.5% which shows that tuning the stepsize is important for PGD. At the same time, FAB achieves a robust accuracy of 0.3% without any need of parameter tuning or rescaling of the classifier. This exemplifies the benefit of the scaling invariance of FAB. Moreover, as a side result this shows that the new loss alone in (Pang et al., 2020) is an ineffective defense.

Runtime comparison: DF and SF are much faster than the other attacks as their primary goal is to find as fast as possible adversarial examples, without emphasis on minimizing their norms, while LRA is rather expensive as noted in the original paper. PGD needs a forward and a backward pass of the network per iteration whereas FAB requires three passes for each iteration. Thus PGD is given 1.5 times more iterations than FAB, so that overall they have same budget of forward/backward passes (and thus runtime). Below we report the runtimes (for 1000 points on MNIST and CIFAR-10, 50 on R-ImageNet) for the attacks as used in the experiments (if not specified otherwise, it includes all the restarts). For PGD and DAA this is the time for evaluating the robust accuracy at 5 thresholds, while for the other methods a single run is sufficient to compute the robust accuracy for all five thresholds. **MNIST**: DAA 11736s, PGD 3825s for l_∞/l_2 and 14106s for l_1, CW 944s, EAD 606s, FAB-10 161s, FAB-100 1613s. **CIFAR-10**: DAA 11625s, PGD 31900s for l_∞/l_2 and 70110s for l_1, CW 3691s, EAD 3398s, FAB-10 1209s, FAB-100 12093s. **R-ImageNet**: DAA 6890s, PGD 4738s for l_∞/l_2 and 24158s for l_1, FAB 2268s for l_∞/l_2 and 3146s for l_1 (note that for l_1 different numbers of restarts/iterations are used on R-ImageNet).

We note that for PGD the robust accuracy for the five thresholds can be computed faster by exploiting the fact that points which are non-robust for a thresholds ϵ are also non-robust for thresholds larger than ϵ. However, even when taking this into account FAB-10 would still be significantly faster than PGD-100 and has better quality on MNIST and CIFAR-10. Moreover, when just considering a fixed number of thresholds, one can stop FAB-attack whenever it finds an adversarial example for the smallest threshold which also leads to a speed-up. However, in real world applications a full picture of robustness as a continuous function of the threshold is the most interesting evaluation scenario.

3.1. Additional results

In the supplementary materials we show how the robust accuracy provided by either PGD or FAB-attack evolves over iterations, when only one start it used. In particular, we compare the two methods when the same number of passes, forward or backward, of the networks are used. One can observe that a few steps are usually sufficient for FAB-attack to achieve good results, often faster than PGD, although there are cases where a higher number of iterations leads to significantly better robust accuracy.

Finally, (Croce & Hein, 2020) use FAB-attack together with other white- and black-box attacks to evaluate the robustness of over 50 classifiers trained with recently proposed adversarial defenses wrt l_∞ and l_2 on different datasets. With fixed hyperparameters, FAB-attack yields the best results in most of the cases on CIFAR-10, CIFAR-100 and ImageNet in both norms, in particular compared to different variations of PGD (with and without a momentum term, with different step sizes and using various losses). This shows again that FAB-attack is very effective for testing the robustness of adversarial defenses.

4. FAB-attack with a large number of classes

The standard algorithm of FAB-attack requires to compute at each iteration the Jacobian matrix of the classifier f wrt the input x and then the closest approximated decision hyperplane. The Jacobian matrix has dimension $K \times d$, recalling that K is the number of classes and d the input dimension. Although this can be in principle obtained with a single backward pass of the network, it becomes computationally expensive on datasets with many classes. Moreover, the memory consumption of FAB-attack increases with K. As a consequence, using FAB-attack in the normal formulation on datasets like ImageNet which has $K = 1000$ classes may be inefficient.

Then, we propose a *targeted* version of our attack which performs at each iteration the projection onto the linearized decision boundary between the original class and a fixed target class. This means that in (8) the hyperplane π_s is not selected via (7) as the closest one to the current iterate but rather $s \equiv t$, with t the target class used. Note that in practice we do not constrain the final outcome of the algorithm to be assigned to class t, but any misclassification is sufficient to have a valid adversarial example. The target class t is selected as the second most likely one according to the score given by the model to the target point, and if k restarts are allowed one can use the classes with the $k + 1$ highest scores as target (excluding the correct one c). In this way, only the gradient of $f_t - f_c : \mathbb{R}^d \to \mathbb{R}$ needs to be computed, which is a cheaper operation than getting the full Jacobian of f and with computational cost independent of the total number of classes.

This targeted version of FAB-attack has been used in (Croce & Hein, 2020) considering the top-10 classes where it yields on CIFAR-10, CIFAR-100 and ImageNet almost always better robust accuracy than normal FAB-attack, which instead is almost always better on MNIST.

5. Conclusion

In summary, our geometrically motivated FAB-attack outperforms in terms of average quality the state-of-the-art attacks, already with a limited computational effort, and works for all l_p-norms in $p \in \{1, 2, \infty\}$ unlike most competitors. Thanks to its scaling invariance and being step size free it is resistant to gradient masking and thus more reliable for assessing robustness than the standard PGD attack.

Acknowledgements

We acknowledge support from the German Federal Ministry of Education and Research (BMBF) through the Tübingen AI Center (FKZ: 01IS18039A). This work was also supported by the DFG Cluster of Excellence "Machine Learning – New Perspectives for Science", EXC 2064/1, project number 390727645, and by DFG grant 389792660 as part of TRR 248.

References

Athalye, A., Carlini, N., and Wagner, D. A. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *ICML*, 2018.

Bastani, O., Ioannou, Y., Lampropoulos, L., Vytiniotis, D., Nori, A., and Criminisi, A. Measuring neural net robustness with constraints. In *NeurIPS*, 2016.

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

Brown, T. B., Mané, D., Roy, A., Abadi, M., and Gilmer, J. Adversarial patch. In *NeurIPS 2017 Workshop on Machine Learning and Computer Security*, 2017.

Carlini, N. and Wagner, D. Towards evaluating the robustness of neural networks. In *IEEE Symposium on Security and Privacy*, 2017a.

Carlini, N. and Wagner, D. Adversarial examples are not easily detected: Bypassing ten detection methods. In *ACM Workshop on Artificial Intelligence and Security*, 2017b.

Chen, P., Sharma, Y., Zhang, H., Yi, J., and Hsieh, C. Ead: Elastic-net attacks to deep neural networks via adversarial examples. In *AAAI*, 2018.

Croce, F. and Hein, M. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *ICML*, 2020.

Croce, F., Rauber, J., and Hein, M. Scaling up the randomized gradient-free adversarial attack reveals overestimation of robustness using established attacks. *International J. of Computer Vision (IJCV)*, 2019.

Engstrom, L., Tran, B., Tsipras, D., Schmidt, L., and Madry, A. A rotation and a translation suffice: Fooling CNNs with simple transformations. In *NeurIPS 2017 Workshop on Machine Learning and Computer Security*, 2017.

Gu, S. and Rigazio, L. Towards deep neural network architectures robust to adversarial examples. In *ICLR Workshop*, 2015.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *CVPR*, pp. 770–778, 2016.

Hein, M. and Andriushchenko, M. Formal guarantees on the robustness of a classifier against adversarial manipulation. In *NeurIPS*, 2017.

Huang, R., Xu, B., Schuurmans, D., and Szepesvari, C. Learning with a strong adversary. In *ICLR*, 2016.

Katz, G., Barrett, C., Dill, D., Julian, K., and Kochenderfer, M. Reluplex: An efficient smt solver for verifying deep neural networks. In *CAV*, 2017.

Krizhevsky, A., Nair, V., and Hinton, G. Cifar-10 (canadian institute for advanced research). 2014. URL http://www.cs.toronto.edu/~kriz/cifar.html.

Kurakin, A., Goodfellow, I. J., and Bengio, S. Adversarial examples in the physical world. In *ICLR Workshop*, 2017.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Valdu, A. Towards deep learning models resistant to adversarial attacks. In *ICLR*, 2018.

Modas, A., Moosavi-Dezfooli, S., and Frossard, P. Sparsefool: a few pixels make a big difference. In *CVPR*, 2019.

Moosavi-Dezfooli, S.-M., Fawzi, A., and Frossard, P. Deepfool: a simple and accurate method to fool deep neural networks. In *CVPR*, pp. 2574–2582, 2016.

Mosbach, M., Andriushchenko, M., Trost, T., Hein, M., and Klakow, D. Logit pairing methods can fool gradient-based attacks. In *NeurIPS 2018 Workshop on Security in Machine Learning*, 2018.

Narodytska, N. and Kasiviswanathan, S. P. Simple black-box adversarial perturbations for deep networks. In *CVPR 2017 Workshops*, 2016.

Pang, T., Xu, K., Dong, Y., Du, C., Chen, N., and Zhu, J. Rethinking softmax cross-entropy loss for adversarial robustness. In *ICLR*, 2020.

Papernot, N., McDonald, P., Wu, X., Jha, S., and Swami, A. Distillation as a defense to adversarial perturbations against deep networks. In *IEEE Symposium on Security & Privacy*, 2016.

Papernot, N., Carlini, N., Goodfellow, I., Feinman, R., Faghri, F., Matyasko, A., Hambardzumyan, K., Juang, Y.-L., Kurakin, A., Sheatsley, R., Garg, A., and Lin, Y.-C. cleverhans v2.0.0: an adversarial machine learning library. preprint, arXiv:1610.00768, 2017.

Rauber, J., Brendel, W., and Bethge, M. Foolbox: A python toolbox to benchmark the robustness of machine learning models. In *ICML Reliable Machine Learning in the Wild Workshop*, 2017.

Su, J., Vargas, D. V., and Kouichi, S. One pixel attack for fooling deep neural networks. *IEEE Transactions on Evolutionary Computation*, 23:828–841, 2019.

Tjeng, V., Xiao, K., and Tedrake, R. Evaluating robustness of neural networks with mixed integer programming. In *ICLR*, 2019.

Tramèr, F. and Boneh, D. Adversarial training and robustness for multiple perturbations. In *NeurIPS*, 2019.

Tsipras, D., Santurkar, S., Engstrom, L., Turner, A., and Madry, A. Robustness may be at odds with accuracy. In *ICLR*, 2019.

Wong, E., Schmidt, F. R., and Kolter, J. Z. Wasserstein adversarial examples via projected sinkhorn iterations. In *ICML*, 2019.

Zheng, S., Song, Y., Leung, T., and Goodfellow, I. J. Improving the robustness of deep neural networks via stability training. In *CVPR*, 2016.

Zheng, T., Chen, C., and Ren, K. Distributionally adversarial attack. In *AAAI*, 2019.