Reliable Evaluation of Adversarial Robustness with an Ensemble of Diverse Parameter-free Attacks

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

Abstract

The field of defense strategies against adversarial attacks has significantly grown over the last years, but progress is hampered as the evaluation of adversarial defenses is often insufficient and thus gives a wrong impression of robustness. Many promising defenses could be broken later on, making it difficult to identify the state-of-the-art. Frequent pitfalls in the evaluation are improper tuning of hyperparameters of the attacks, gradient obfuscation or masking. In this paper we first propose two extensions of the PGD-attack overcoming failures due to suboptimal step size and problems of the objective function. We then combine our novel attacks with two complementary existing ones to form a parameter-free, computationally affordable and user-independent ensemble of attacks to test adversarial robustness. We apply our ensemble to over 50 models from papers published at recent top machine learning and computer vision venues. In all except one of the cases we achieve lower robust test accuracy than reported in these papers, often by more than 10%, identifying several broken defenses.

1. Introduction

Adversarial samples, small perturbations of the input, with respect to some distance measure, which change the decision of a classifier, are a problem for safe and robust machine learning. In particular, they are a major concern when it comes to safety-critical applications. In recent years many defenses have been proposed but with more powerful or adapted attacks most of them could be broken (Carlini & Wagner, 2017b; Athalye et al., 2018; Mosbach et al., 2018). Adversarial training (Madry et al., 2018) is one of the few approaches which could not be defeated so far. Recent variations are using other losses (Zhang et al., 2019b) and boost robustness via generation of additional training data (Carmon et al., 2019; Alayrac et al., 2019) or pre-training (Hendrycks et al., 2019). Another line of work are provable defenses, either deterministic (Wong et al., 2018; Croce et al., 2019a; Mirman et al., 2018; Gowal et al., 2019a) or based on randomized smoothing (Li et al., 2019; Lecuyer et al., 2019; Cohen et al., 2019). However, these are not yet competitive with the empirical robustness of adversarial training for datasets like CIFAR-10 with large perturbations.

Due to the many broken defenses, the field is currently in a state where it is very difficult to judge the value of a new defense without an independent test. This limits the progress as it is not clear how to distinguish bad from good ideas. A seminal work to mitigate this issue are the guidelines for assessing adversarial defenses by (Carlini et al., 2019). However, as we see in our experiments, even papers trying to follow these guidelines can fail in obtaining a proper evaluation. In our opinion the reason is that at the moment there is no protocol which works reliably and autonomously, and does not need the fine-tuning of parameters for every new defense. Such protocol is what we aim at in this work.

The most popular method to test adversarial robustness is the PGD (Projected Gradient Descent) attack (Madry et al., 2018), as it is computationally cheap and performs well in many cases. However, it has been shown that even PGD can fail (Mosbach et al., 2018; Croce et al., 2019b) leading to significant overestimation of robustness: we identify i) the fixed step size and ii) the widely used cross-entropy loss as two reasons for potential failure. As remedies we propose i) a new gradient-based scheme, Auto-PGD, which does not require a step size to be chosen (Sec. 3), and ii) an alternative loss function (Sec. 4). These novel tools lead to two variants of PGD whose only free parameter is the number of iterations, while everything else is adjusted automatically: this is the first piece of the proposed evaluation protocol.

Another cause of poor evaluations is the lack of diversity among the attacks used, as most papers rely solely on the results given by PGD or weaker versions of it like FGSM (Goodfellow et al., 2015). Of different nature are for example two existing attacks: the white-box FAB-attack (Croce & Hein, 2019) and the black-box Square Attack

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

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

(Andriushchenko et al., 2019). Importantly, these methods have a limited amount of parameters which generalize well across classifiers and datasets. In Sec. 5, we combine our two new versions of PGD with FAB and Square Attack to form a parameter-free, computationally affordable and user-independent ensemble of complementary attacks to estimate adversarial robustness, named *AutoAttack*.

We test *AutoAttack* in a large-scale evaluation (Sec. 6) on over 50 classifiers from 35 papers proposing robust models, including randomized defenses, from recent leading conferences. Although using only five restarts for each of the three white-box attacks contained in *AutoAttack*, in all except two cases the robust test accuracy obtained by *AutoAttack* is lower than the one reported in the original papers (our slightly more expensive *AutoAttack+* is better in all but one case). For 13 models we reduce the robust accuracy by more than 10% and identify several broken defenses.

We do not argue that *AutoAttack* is the ultimate adversarial attack but rather that it should become the minimal test for any new defense, since it reliably reaches good performance in all tested models, *without any hyperparameter tuning and at a relatively low computational cost*. At the same time our large-scale evaluation identifies the current state-of-the-art and which of the recent ideas are actually effective.

2. Adversarial examples and PGD

Let $g : \mathcal{D} \subseteq \mathbb{R}^d \longrightarrow \mathbb{R}^K$ be a K-class classifier taking decisions according to $\arg\max_{k=1,\dots,K} g_k(\cdot)$ and $x_{\text{orig}} \in \mathbb{R}^d$ a point which is correctly classified by g as c. Given a metric $d(\cdot,\cdot)$ and $\epsilon > 0$, the threat model (feasible set of the attack) is defined as $\{z \in \mathcal{D} \,|\, d(x_{\text{orig}}, z) \leq \epsilon\}$. Then z is an adversarial sample for g at x_{orig} wrt the threat model if

$$\arg\max_{k=1,\dots,K} g_k(z) \neq c, \quad d(x_{\text{orig}}, z) \leq \epsilon \quad \text{and} \quad z \in \mathcal{D}.$$

To find z it is common to define some surrogate function L such that solving the constrained optimization problem

$$\max_{z \in \mathcal{D}} L(g(z), c) \quad \text{such that} \quad \gamma(x_{\text{orig}}, z) \leq \epsilon, \; z \in \mathcal{D} \quad (1)$$

enforces z not to be assigned to class c. In image classification, the most popular threat models are based on l_p-distances, i.e. $d(x, z) := \|z - x\|_p$, and $\mathcal{D} = [0, 1]^d$. Since the projection on the l_p-ball for $p \in \{2, \infty\}$ is available in closed form, Problem (1) can be solved with *Projected Gradient Descent* (PGD). Given $f : \mathbb{R}^d \longrightarrow \mathbb{R}$, $\mathcal{S} \subset \mathbb{R}^d$ and the problem $\max_{x \in \mathcal{S}} f(x)$, the iterations of PGD are defined for $k = 1, \dots, N_{\text{iter}}$ as $x^{(k+1)} = P_{\mathcal{S}}\left(x^{(k)} + \eta^{(k)} \nabla f(x^{(k)})\right)$, where $\eta^{(k)}$ is the step size at iteration k and $P_{\mathcal{S}}$ is the projection onto $\mathcal{S}$. Using the cross-entropy (CE) loss as objective L, (Kurakin et al., 2017; Madry et al., 2018) introduced the so-called **PGD-attack**, which is currently the most popular white-box attack. In their formulation $\eta^{(k)} = \eta$ for every k, i.e. the step size is fixed, and as initial point $x^{(0)}$ either x_{orig} or $x_{\text{orig}} + \zeta$ is used, where ζ is randomly sampled such that $x^{(0)}$ satisfies the constraints. Moreover, steepest descent is done according to the norm of the threat model (e.g. for l_∞ the sign of the gradient is used).

Algorithm 1 APGD

Input: f, S, $x^{(0)}$, η, N_{iter}, $W = \{w_0, \dots, w_n\}$
Output: x_{max}, f_{max}
$x^{(1)} \leftarrow P_{\mathcal{S}}\left(x^{(0)} + \eta \nabla f(x^{(0)})\right)$
$f_{\text{max}} \leftarrow \max\{f(x^{(0)}), f(x^{(1)})\}$
$x_{\text{max}} \leftarrow x^{(0)}$ **if** $f_{\text{max}} \equiv f(x^{(0)})$ **else** $x_{\text{max}} \leftarrow x^{(1)}$
for $k = 1$ **to** $N_{\text{iter}} - 1$ **do**
  $z^{(k+1)} \leftarrow P_{\mathcal{S}}\left(x^{(k)} + \eta \nabla f(x^{(k)})\right)$
  $x^{(k+1)} \leftarrow P_{\mathcal{S}}\Big(x^{(k)} + \alpha(z^{(k+1)} - x^{(k)}) + (1 - \alpha)(x^{(k)} - x^{(k-1)})\Big)$
  if $f(x^{(k+1)}) > f_{\text{max}}$ **then**
    $x_{\text{max}} \leftarrow x^{(k+1)}$ and $f_{\text{max}} \leftarrow f(x^{(k+1)})$
  end if
  if $k \in W$ **then**
    if Condition 1 **or** Condition 2 **then**
      $\eta \leftarrow \eta/2$ and $x^{(k+1)} \leftarrow x_{\text{max}}$
    end if
  end if
end for

3. Auto-PGD: A budget-aware step size-free variant of PGD

We identify three weaknesses in the standard formulation of the PGD-attack and how it is used in the context of adversarial robustness. First, the **fixed step size** is suboptimal, as even for convex problems this does not guarantee convergence, and the performance of the algorithm is highly influenced by the choice of its value, see e.g. (Mosbach et al., 2018). Second, the overall scheme is in general **agnostic of the budget** given to the attack: as we show, the loss plateaus after a few iterations, except for extremely small step sizes, which however do not translate into better results. As a consequence, judging the strength of an attack by the number of iterations is misleading, see also (Carlini et al., 2019). Finally, the algorithm is **unaware of the trend**, i.e. does not consider whether the optimization is evolving successfully and is not able to react to this.

3.1. Auto-PGD (APGD) algorithm

In our automatic scheme we aim at fixing these issues. The main idea is to partition the available N_{iter} iterations in an

initial exploration phase, where one searches the feasible set for *good* initial points, and an exploitation phase, during which one tries to maximize the knowledge so far accumulated. The transition between the two phases is managed by progressively reducing the step size. In fact, a large step size allows to move quickly in $\mathcal{S}$, whereas a smaller one more eagerly maximizes the objective function locally. However, the reduction of the step size is not a priori scheduled, but rather governed by the trend of the optimization: if the value of the objective grows sufficiently fast, then the step size is most likely proper, otherwise it is reasonable to reduce it. While the update step in APGD is standard, what distinguishes our algorithm from usual PGD is the choice of the step size across iterations, which is adapted to the overall budget and to the progress of the optimization, and that, once the step size is reduced, the maximization restarts from the best point so far found. We summarize our scheme in Algorithm 1 and analyze the main features in the following.

Gradient step: The update of APGD follows closely the classic algorithm and only adds a momentum term. Let $\eta^{(k)}$ be the step size at iteration k, then the update step is

$$\begin{aligned} z^{(k+1)} &= P_{\mathcal{S}}\left(x^{(k)} + \eta^{(k)} \nabla f(x^{(k)})\right) \\ x^{(k+1)} &= P_{\mathcal{S}}\Big(x^{(k)} + \alpha \cdot (z^{(k+1)} - x^{(k)}) \\ &\qquad + (1-\alpha) \cdot (x^{(k)} - x^{(k-1)})\Big), \end{aligned} \tag{2}$$

where $\alpha \in [0, 1]$ (we use $\alpha = 0.75$) regulates the influence of the previous update on the current one. Since in the early iterations of APGD the step size is particularly large, we want to keep a bias from the previous steps.

Step size selection: We start with step size $\eta^{(0)}$ at iteration 0 (we fix $\eta^{(0)} = 2\epsilon$), and given a budget of N_{iter} iterations, we identify checkpoints $w_0 = 0, w_1, \dots, w_n$ at which the algorithm decides whether it is necessary to halve the current step size. We have two conditions:

1. $\sum_{i=w_{j-1}}^{w_j - 1} \mathbf{1}_{f(x^{(i+1)}) > f(x^{(i)})} < \rho \cdot (w_j - w_{j-1})$,
2. $\eta^{(w_{j-1})} \equiv \eta^{(w_j)}$ and $f_{\text{max}}^{(w_{j-1})} \equiv f_{\text{max}}^{(w_j)}$,

where $f_{\text{max}}^{(k)}$ is the highest objective value found in the first k iterations. If one of the conditions is true, then step size at iteration $k = w_j$ is halved and $\eta^{(k)} := \eta^{(w_j)}/2$ for every $k = w_j + 1, \dots, w_{j+1}$.
Condition 1: counts in how many cases since the last checkpoint w_{j-1} the update step has been successful in increasing f. If this happened for at least a fraction ρ of the total update steps, then the step size is kept as the optimization is proceeding properly (we use $\rho = 0.75$).
Condition 2: holds true if the step size was not reduced at the last checkpoint *and* there has been no improvement in the best found objective value since the last checkpoint. This prevents getting stuck in potential cycles.

Restarts from the best point: If at a checkpoint w_j the step size gets halved, then we set $x^{(w_j+1)} := x_{\text{max}}$, that is we restart at the point attaining the highest objective f_{max} so far. This makes sense as reducing η leads to a more localized search, and this should be done in a neighborhood of the current best candidate solution.

Exploration vs exploitation: We want the algorithm to transit gradually from exploring the whole feasible set $\mathcal{S}$ to a local optimization. This transition is regulated by progressively reducing the step size and by the choice of when to decrease it, i.e. the checkpoints w_j. In practice, we want to allow a relatively long initial exploration stage and then possibly update the step size more often moving toward exploitation. In fact, with smaller step sizes the improvements in the objective function are likely more frequent but also of smaller magnitude, while the importance of taking advantage of the whole input space is testified by the success of random restarts in the usual PGD-attack. We fix the checkpoints as $w_j = \lceil p_j N_{\text{iter}} \rceil \leq N_{\text{iter}}$, with $p_j \in [0, 1]$ defined as $p_0 = 0$, $p_1 = 0.22$ and

$$p_{j+1} = p_j + \max\{p_j - p_{j-1} - 0.03, 0.06\}.$$

Note that the period length $p_{j+1} - p_j$ is reduced in each step by 0.03 but they have at least a minimum length of 0.06.

While the proposed scheme has a few parameters which could be adjusted, we fix them to the values indicated so that the **only free variable is the budget** N_{iter}.

3.2. Comparison of APGD to usual PGD

We compare our APGD to PGD with Momentum in terms of achieved CE loss and robust accuracy, focusing here on l_∞-attacks with perturbation size ϵ. We attack the robust models on MNIST and CIFAR-10 from (Madry et al., 2018) and (Zhang et al., 2019b). We run 1000 iterations of PGD with Momentum with step sizes ϵ/t with $t \in \{0.5, 1, 2, 4, 10, 25, 100\}$, and APGD with a budget of $N_{\text{iter}} \in \{25, 50, 100, 200, 400, 1000\}$ iterations. In Figure 1 we show the evolution of the current best average cross-entropy loss and robust accuracy (i.e. the percentage of points for which the attack could not find an adversarial example) for 1000 points of the test as a function of iterations. In all cases APGD achieves the highest loss (higher is better as it is a maximization problem) and this holds for any budget of iterations. Similarly, APGD attains always the lowest (better) robust accuracy and thus is the stronger adversarial attack on these models (see supplements for a comparison across all models and different losses). One can observe the adaptive behaviour of APGD: when the budget of iterations is larger the value of the objective (the CE loss)

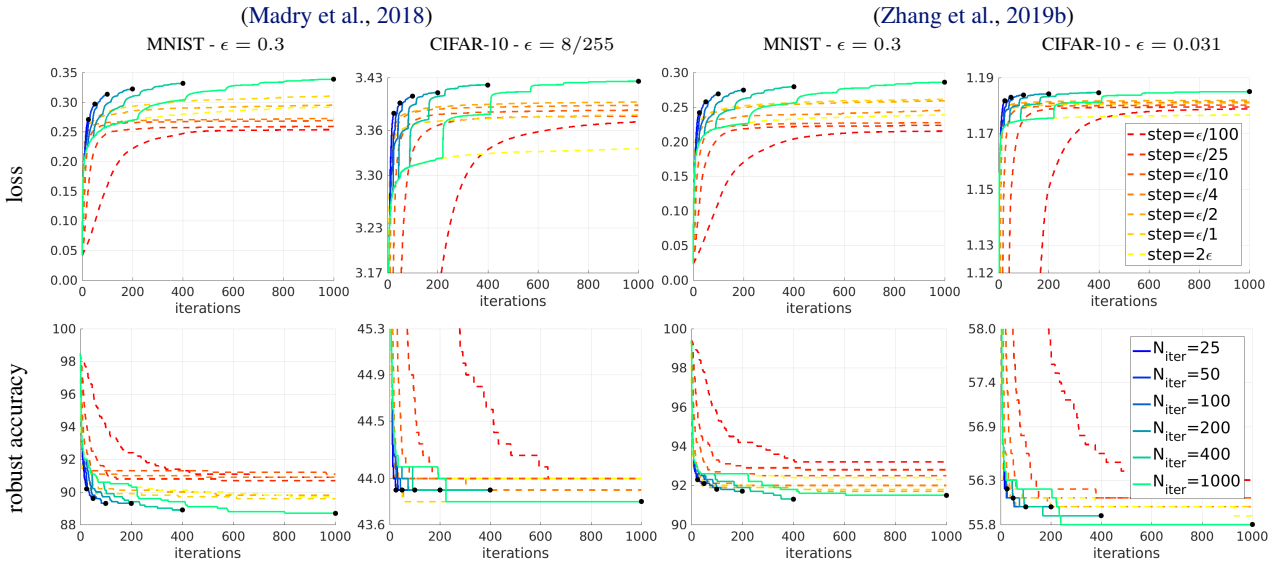

Figure 1. **PGD with Momentum vs APGD:** best cross-entropy loss (top) and robust accuracy (bottom) obtained so far as function of iterations for the models of (Madry et al., 2018) and TRADES (Zhang et al., 2019b) for PGD with a momentum term ($\alpha = 0.75$, as used in APGD) (dashed lines) with different fixed step sizes (always 1000 iterations) and APGD (solid lines) with different budgets of iterations. APGD outperforms PGD with Momentum for every budget of iterations in achieved loss and almost always in robust accuracy.

increases more slowly, but reaches higher values in the end. This is due to the longer exploration phase, which sacrifices smaller improvements to finally get better results. In contrast, the runs of PGD with Momentum tend to plateau at suboptimal values, regardless of the choice of the step size. An analogous comparison of APGD to PGD without momentum can be found in the supplement.

4. An alternative loss

If x has correct class y, the cross-entropy loss at x is

$$\text{CE}(x, y) = -\log p_y = -z_y + \log\Big(\sum_{j=1}^{K} e^{z_j}\Big), \qquad (3)$$

with $p_i = e^{z_i} / \sum_{j=1}^{K} e^{z_j}$, $i = 1, \dots, K$, which is invariant to shifts of the logits z but not to rescaling, similarly to its gradient wrt x, given by

$$\nabla_x \text{CE}(x, y) = (-1 + p_y)\, \nabla_x z_y + \sum_{i \neq y} p_i \nabla_x z_i. \qquad (4)$$

If $p_y \approx 1$ and consequently $p_i \approx 0$ for $i \neq y$, then $\nabla_x \text{CE}(x, y) \approx \mathbf{0}$ and finite arithmetic yields $\nabla_x \text{CE}(x, y) = \mathbf{0}$ (this phenomenon of gradient vanishing is observed in (Carlini & Wagner, 2017a)). Notice that one can achieve $p_y \approx 1$ with a classifier $h = \alpha g$ equivalent to g (i.e. they take the same decision for every x) but rescaled by a constant $\alpha > 0$. To exemplify how this can lead to overestimation of robustness, we run 100 iterations of the

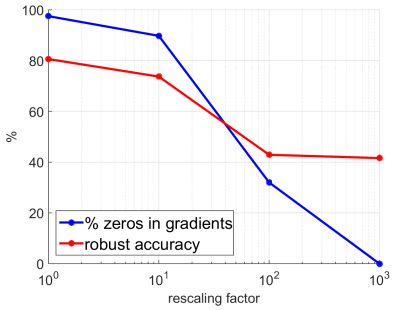

Figure 2. Percentage of zeros in the gradients and robust accuracy, computed by PGD on the CE loss, of the classifiers g/α, where g is the CIFAR-10 model of (Atzmon et al., 2019) and α a rescaling factor. The performance of PGD depends on the scale of the logits.

l_∞ PGD-attack on the CE loss on the CIFAR-10 model from (Atzmon et al., 2019), with $\epsilon = 0.031$, dividing the logits by a factor $\alpha \in \{1, 10^1, 10^2, 10^3\}$. In Figure 2 we show the fraction of entries in the gradients of g/α (g is the original model) equal to zero and the robust accuracy achieved by the attack in dependency on α (we use 1000 test points, the gradient statistic is computed for correctly classified points). Without rescaling ($\alpha = 1$) the gradient vanishes almost for every coordinate, so that PGD is ineffective, but simply rescaling the logits is sufficient to get a much more accurate robustness assessment (see also supplement).

(Carlini & Wagner, 2017a) proposed the CW loss

$$\text{CW}(x, y) = -z_y + \max_{i \neq y} z_i. \qquad (5)$$

In contrast to the CE loss it has a direct interpretation in terms of the decision of the classifier, in particular if an adversarial example exists, then the global maximum of the CW loss is positive. However, the CW loss is not scaling invariant and thus again an extreme rescaling could in principle be used to induce gradient masking.

4.1. Difference of Logits Ratio Loss

We propose the **Difference of Logits Ratio (DLR)** loss which is both shift and rescaling invariant:

$$\text{DLR}(x, y) = -\frac{z_y - \max_{i \neq y} z_i}{z_{\pi_1} - z_{\pi_3}}, \tag{6}$$

where π is the ordering of the components of z in decreasing order. The required shift-invariance of the loss is achieved by having a difference of logits also in the denominator. Maximizing DLR wrt x allows to find a point classified not in class y (DLR is positive only if $\text{argmax}_i z_i \neq y$) and, once that is achieved, minimizes the score of class y compared to that of the other classes. If x is correctly classified we have $\pi_1 \equiv y$, so that $\text{DLR}(x, y) = -\frac{z_y - z_{\pi_2}}{z_y - z_{\pi_3}}$ and $\text{DLR}(x, y) \in [-1, 0]$. The role of the normalization $z_{\pi_1} - z_{\pi_3}$ is to push z_{π_2} to $z_y = z_{\pi_1}$ as it prefers points for which $z_y \approx z_{\pi_2} > z_{\pi_3}$ and thus is biased towards changing the decision.

4.2. DLR loss versus CW and CE loss and evaluation of APGD versus PGD

We compare the three losses when optimized via PGD, PGD with Momentum (same momentum as in APGD) and APGD with the same budget on all deterministic models trained for l_∞-robustness used in the main experiments (see Sections 6 and Appendix) using CE, CW and DLR loss (the complete results are reported in the supplementary material). For both PGD and PGD with Momentum we use three step sizes ($\epsilon/10$, $\epsilon/4$, 2ϵ, with ϵ the bound on the norm of the perturbations). The version of standard PGD achieving most often the lowest robust accuracy is for all three losses PGD with Momentum and step size $\epsilon/4$, which gives average robust accuracy (over all models) of 54.84%, 50.42% and 50.47% on the CE, CW and DLR loss respectively. In the same metric, APGD achieves 54.00%, 49.46%, 48.53%. Moreover, when compared on every classifier individually, the best attack (over all PGD versions and APGD) with the CW loss can be up to 21% worse than the best one with the DLR loss, while it is never better by more than 5%, which means that our loss is more stable. Finally, APGD outperforms the best among the 6 versions of PGD on 32 of the 43 models for the CE loss, CW 37/43 and DLR 35/43 and the models where APGD is worse are mainly the ones where the extreme stepsize 2ϵ is optimal as the defenses lead to gradient masking/obfuscation (further details in supplements). In total these experiments show: i) the DLR loss improves upon CE and is comparable to the CW loss, but with less severe failure cases, and ii) APGD outperforms PGD/PGD with Momentum on all three losses consistently.

We run a similar comparison for the models trained to be robust wrt l_2 and report the results in in the supplementary material. The relative difference among the losses in this case is minimal, and the performance of the 6 versions of PGD and APGD are similar, although APGD still yields most often the best results for all the losses.

5. *AutoAttack*: an ensemble of parameter-free attacks

We combine our two parameter-free versions of PGD, APGD_{CE} and APGD_{DLR}, with two existing complementary attacks, FAB (Croce & Hein, 2019) and Square Attack (Andriushchenko et al., 2019), to form the ensemble *AutoAttack*, which is automatic in the sense that it does not require to specify any free parameters[1].

A key property of *AutoAttack* is the diversity of its components: while APGD is a white-box attack aiming at any adversarial example within an l_p-ball, FAB minimizes the norm of the perturbation necessary to achieve a misclassification, and, although it relies on the gradient of the logits, it appears to be effective also on models affected by gradient masking as shown in (Croce & Hein, 2019). On the other hand, Square Attack is a score-based black-box attack for norm bounded perturbations which uses random search and does not exploit any gradient approximation. It outperforms other black-box attacks in terms of query efficiency and success rate and has been shown to be even competitive with white-box attacks (Andriushchenko et al., 2019). Both methods have few parameters which generalize well across models and datasets, so that we will keep them fixed for all experiments. The diversity within the ensemble helps for two reasons: first, there exist classifiers for which some of the attacks dramatically fail, but always at least one of them works well. Second, on the same model diverse attacks might have similar robust accuracy but succeed on different points: then considering the worst case over all attacks, as we do for *AutoAttack*, improves the performance.

Each attack in *AutoAttack* gets a relatively limited computational budget: for APGD_{CE}, APGD_{DLR} and FAB we use for each 100 iterations and 5 random restarts, for Square Attack one run with 5000 queries. While the runtime depends on the model, its robustness and even the framework of the target network, APGD is the fastest attack, as it requires only one forward and one backward pass per iteration. The computational budget of *AutoAttack* is similar to what has

[1] *AutoAttack* is available at https://github.com/fra31/auto-attack.

been used, on average, in the evaluation of the defenses considered. The hyperparameters of all attacks are fixed for all experiments across datasets, models and norms (see supplementary material). In Sec. 6 we show that *AutoAttack* evaluates adversarial robustness reliably and cost-efficiently despite its limited budget and running fully automatic, without any hyperparameter tuning.

6. Experiments

In order to test the performance of *AutoAttack*, but also the individual performances of APGD_{CE} and APGD_{DLR}, we evaluate the adversarial robustness in the l_∞- and l_2-threat models of over 50 models of 35 defenses from recent conferences like ICML, NeurIPS, ICLR, ICCV, CVPR, using MNIST, CIFAR-10, CIFAR-100 and ImageNet as datasets. We report first results for deterministic defenses (additional ones with other thresholds ϵ in the supplementary materials) and then for randomized ones, i.e. classifiers which have a stochastic component. *AutoAttack* improves almost all evaluations, showing that its good performance generalizes very well across datasets, models and threat models with the same hyperparameters.

Deterministic defenses: In Tables 1 (and in the Appendix)we report the results on 49 models, 43 trained for l_∞- and 6 for l_2-robustness, from recent defense papers (for some of them multiple networks are considered, possibly on more datasets and norms). When possible we used the originals models (which are either publicly available or we obtained them via personal communication from the authors). Otherwise we retrained the models with the code released by the authors. Further details about the models and papers can be found in the Appendix. For each classifier we report the clean accuracy and robust accuracy, at the ϵ specified in the table, on the whole test set (except for ImageNet where we use 1000 points from the validation set) obtained by the individual attacks APGD_{CE}, APGD_{DLR}, FAB and Square Attack, together with our ensemble *AutoAttack*, which counts as a success every point on which *at least one* of the four attacks finds an adversarial example (worst case evaluation). Additionally, we provide the *reported* robust accuracy of the respective papers (please note that in some cases their statistics are computed on a subset of the test set) and the difference between our robust accuracy and the reported one. The reduction is highlighted in red in the last column of Table 1 if it is negative (we get a lower robust accuracy). Finally, we boldface the attack which obtains the best individual robust accuracy and underline those which achieve a robust accuracy lower than reported.

Notably, in all but two cases *AutoAttack* achieves a lower robust accuracy than reported in the original papers, and the improvement is larger than 10% in 13 out of 49 cases, larger than 30% in 8 cases (*AutoAttack* achieves also significantly lower robust accuracy on the few models on CIFAR-100 and ImageNet). Thus *AutoAttack* would almost always have provided a better estimate of the robustness of the models than in the original evaluation, without any adaptation to the specific defense. In the only cases where it does not reduce the reported robust accuracy it is at most only 0.62% far from it, and these results have been obtained with a variant of PGD with 180 restarts and 200 iterations (see (Alayrac et al., 2019; Qin et al., 2019)), which is way more expensive than our evaluation (see below and supplementary material).

In most of the cases more than one of the attacks included in *AutoAttack* achieves a lower robust accuracy than reported (APGD_{CE} improves the reported evaluation in 34/49 cases, APGD_{DLR} in 38/49, FAB in 39/49 and Square Attack in 17/49, but 9/9 on MNIST). FAB most often attains the best result for CIFAR-10, CIFAR-100 and ImageNet, and Square Attack on MNIST. However, APGD_{DLR} is the most reliable one as it has the least severe failure which we define as the largest difference in robust accuracy to the best performing attack (maximal difference less than 11%, compared to 85% for APGD_{CE}, 59% for FAB and 70% for Square Attack). Thus our new DLR loss is able to resist gradient masking.

Randomized defenses: Another line of adversarial defenses relies on adding to a classifier some stochastic component. In this case the output (and thus the decision) of the model might change across different runs for the same input. Thus we compute in Table 2 the mean (standard deviation in brackets) of our statistics over 5 runs. Moreover the results of *AutoAttack* are given considering, for each point, the attack performing better on average, i.e. across 5 runs. In order to adapt *AutoAttack* to handle randomized defenses we adopt two strategies. For the models from (Grathwohl et al., 2020), we attack the model named JEM-0 with the standard version of our ensemble, since the stochastic component has little influence, and then reuse the same adversarial examples on the other models (JEM-1 and JEM-10). For the other classifiers, the direction for the update step in APGD is computed taking the average of 20 computations of the gradient at the same point (known as Expectation over Transformation (Athalye et al., 2018)). In this case we use only 1 random start instead of 5. We do not run FAB here since it returns points on or very close to the decision boundary, so that even a small variation in the classifier is likely to undo the adversarial change, as confirmed by the bad performance on the models from (Grathwohl et al., 2020). We modify Square Attack so that it accepts an update if it reduces the target loss on average over 20 forward passes. As this costs more time we use only 1000 iterations for Square Attack. Table 2 shows that *AutoAttack* achieves always lower robust accuracy than reported in the respective papers, with APGD_{CE} being the best performing attack, closely followed by APGD_{DLR}. In 7 out of 9 cases the improvement is significant, larger than 10% (and in 3/9

Table 1. **Robustness evaluation of adversarial defenses by *AutoAttack*.** We report clean test accuracy, the robust accuracy of the individual attacks as well as the combined one of *AutoAttack* (AA column). We also provide the robust accuracy reported in the original papers and compute the difference to the one of *AutoAttack*. If negative (in red) *AutoAttack* provides lower (better) robust accuracy.

#	paper	clean	$APGD_{CE}$	$APGD_{DLR}$	FAB	Square	AA	reported	reduct.
CIFAR-10 - l_∞ - $\epsilon = 8/255$									
1	(Carmon et al., 2019)	89.69	61.47	60.64	**60.62**	66.63	59.65	62.5	-2.85
2	(Alayrac et al., 2019)	86.46	59.86	62.03	58.20	66.37	56.92	56.30	0.62
3	(Hendrycks et al., 2019)	87.11	57.00	56.96	**55.40**	61.99	54.99	57.4	-2.41
4	(Rice et al., 2020)	85.34	56.76	55.72	**54.13**	61.37	53.60	58	-4.40
5	(Qin et al., 2019)	86.28	55.45	55.46	53.76	60.01	53.07	52.81	0.26
6	(Engstrom et al., 2019)	87.03	51.52	52.62	**50.38**	58.00	49.58	53.29	-3.71
7	(Kumari et al., 2019)	87.80	51.56	51.68	**49.83**	58.20	49.40	53.04	-3.64
8	(Mao et al., 2019)	86.21	49.39	50.33	**48.31**	56.98	47.66	50.03	-2.37
9	(Ding et al., 2020)	84.36	49.36	50.32	47.25	55.49	45.57	47.18	-1.61
10	(Zhang et al., 2019a)	87.20	45.91	47.33	**45.62**	55.07	45.06	47.98	-2.92
11	(Madry et al., 2018)	87.14	**44.56**	46.03	45.37	53.10	44.29	47.04	-2.75
12	(Pang et al., 2020)	80.89	55.91	44.56	**44.44**	49.73	43.78	55.0	-11.22
13	(Wong et al., 2020)	83.34	45.60	46.64	**44.03**	53.32	43.38	46.06	-2.68
14	(Shafahi et al., 2019)	86.11	43.30	44.56	**42.88**	51.95	41.58	46.19	-4.61
15	(Zhang & Wang, 2019)	89.98	62.03	48.96	**40.84**	59.12	38.78	60.6	-21.82
16	(Moosavi-Dezfooli et al., 2019)	83.11	41.59	40.29	**39.05**	47.69	38.67	41.4	-2.73
17	(Zhang & Xu, 2020)	90.25	69.36	49.43	**40.60**	66.87	38.57	68.7	-30.13
18	(Kim & Wang, 2020)	91.51	54.80	48.41	**38.88**	61.31	36.10	57.23	-21.13
19	(Jang et al., 2019)	78.91	37.40	37.01	**35.49**	44.33	35.09	37.40	-2.31
20	(Moosavi-Dezfooli et al., 2019)	80.41	36.53	35.47	**34.07**	43.45	33.76	36.3	-2.54
21	(Wang & Zhang, 2019)	92.80	57.19	40.69	**33.57**	64.32	30.96	58.6	-27.64
22	(Wang & Zhang, 2019)	92.82	67.77	36.72	**31.92**	66.67	29.07	66.9	-37.83
23	(Mustafa et al., 2019)	89.16	4.48	4.54	**1.10**	33.91	0.55	32.32	-31.77
24	(Chan et al., 2020)	93.79	1.90	**1.20**	15.58	71.76	0.18	15.5	-15.32
25	(Pang et al., 2020)	93.52	85.58	0.49	**0.03**	35.83	0.00	31.4	-31.40
CIFAR-10 - l_∞ - $\epsilon = 0.031$									
1	(Zhang et al., 2019b)	84.92	55.08	54.04	**53.82**	59.48	53.18	56.43	-3.25
2	(Atzmon et al., 2019)	81.30	78.94	44.50	**40.79**	47.99	40.61	43.17	-2.56
3	(Xiao et al., 2020)	79.28	32.38	31.27	79.28	**20.44**	17.99	52.4	-34.41
CIFAR-100 - l_∞ - $\epsilon = 8/255$									
1	(Hendrycks et al., 2019)	59.23	32.83	31.68	**29.09**	34.19	28.63	33.5	-4.87
2	(Rice et al., 2020)	53.83	20.32	20.20	**19.27**	23.75	19.02	28.1	-9.08
MNIST - l_∞ - $\epsilon = 0.3$									
1	(Zhang et al., 2020)	98.38	94.58	94.82	95.61	**93.97**	93.95	96.38	-2.43
2	(Zhang et al., 2019b)	99.48	93.14	93.89	94.14	**92.99**	92.76	95.60	-2.84
3	(Gowal et al., 2019a)	98.34	93.81	93.87	94.76	**92.81**	92.75	93.88	-1.13
4	(Ding et al., 2020)	98.95	93.51	93.86	94.26	**91.39**	91.35	92.59	-1.24
5	(Atzmon et al., 2019)	99.35	98.79	94.54	94.15	**90.86**	90.85	97.35	-6.50
6	(Madry et al., 2018)	98.53	89.40	89.74	92.22	**88.58**	88.43	89.62	-1.19
7	(Jang et al., 2019)	98.47	92.45	92.15	93.24	**88.00**	87.99	94.61	-6.62
8	(Wong et al., 2020)	98.50	84.74	85.39	87.36	**83.07**	82.88	88.77	-5.89
9	(Taghanaki et al., 2019)	98.86	23.83	**0.00**	0.02	**0.00**	0.00	64.25	-64.25
ImageNet - l_∞ - $\epsilon = 4/255$									
1	(Engstrom et al., 2019)	63.4	30.9	32.0	**28.8**	46.8	28.0	33.38	-5.38
CIFAR-10 - l_2 - $\epsilon = 0.5$									
1	(Augustin et al., 2020)	91.08	74.65	74.94	74.13	83.10	73.27	73.27	0.00
2	(Engstrom et al., 2019)	90.83	69.60	70.20	**69.54**	80.92	69.29	70.11	-0.82
3	(Rice et al., 2020)	88.67	68.53	68.95	**68.03**	79.01	67.76	71.6	-3.84
4	(Rony et al., 2019)	89.05	**66.57**	67.02	66.81	78.05	66.49	67.6	-1.11
5	(Ding et al., 2020)	88.02	66.19	66.53	66.43	76.99	66.13	66.18	-0.05
ImageNet - l_2 - $\epsilon = 3$									
1	(Engstrom et al., 2019)	55.3	31.5	30.9	**28.9**	46.6	28.6	35.09	-6.49

Table 2. **Robustness evaluation of randomized l_∞-adversarial defenses by *AutoAttack*.** We report the clean test accuracy (mean and standard deviation over 5 runs) and the robust accuracy of the individual attacks as well as the combined on of *AutoAttack* (again over 5 runs). We also provide the robust accuracy reported in the respective papers and compute the difference to the one of *AutoAttack* (negative means that *AutoAttack* is better). The statistics of our attack are computed on the whole test set except for the ones of (Yang et al., 2019), which are on 1000 test points due to the computational cost of this defense. The ϵ is the same as used in the papers.

#	paper	model	clean	APGD_{CE}	APGD_{DLR}	FAB	Square	*AutoAttack*	report.	reduct.
CIFAR-10 - $\epsilon = 8/255$										
1	(Wang et al., 2019)	En_5RN	82.39 (0.14)	**48.81**	49.37	-	78.61	45.56 (0.20)	51.48	-5.9
2	(Yang et al., 2019)	with AT	84.9 (0.6)	**30.1**	31.9	-	-	26.3 (0.85)	52.8	-26.5
3	(Yang et al., 2019)	pure	87.2 (0.3)	**21.5**	24.3	-	-	18.2 (0.82)	40.8	-22.6
4	(Grathwohl et al., 2020)	JEM-10	90.99 (0.03)	**11.69**	15.88	63.07	79.32	9.92 (0.03)	47.6	-37.7
5	(Grathwohl et al., 2020)	JEM-1	92.31 (0.04)	**9.15**	13.85	62.71	79.25	8.15 (0.05)	41.8	-33.6
6	(Grathwohl et al., 2020)	JEM-0	92.82 (0.05)	**7.19**	12.63	66.48	73.12	6.36 (0.06)	19.8	-13.4
CIFAR-10 - $\epsilon = 4/255$										
1	(Grathwohl et al., 2020)	JEM-10	91.03 (0.05)	**49.10**	52.55	78.87	89.32	47.97 (0.05)	72.6	-24.6
2	(Grathwohl et al., 2020)	JEM-1	92.34 (0.04)	**46.08**	49.71	78.93	90.17	45.49 (0.04)	67.1	-21.6
3	(Grathwohl et al., 2020)	JEM-0	92.82 (0.02)	**42.98**	47.74	82.92	89.52	42.55 (0.07)	50.8	-8.2

cases larger than 25%). Thus *AutoAttack* is also suitable for the evaluation of randomized defenses.

6.1. *AutoAttack*+: Improving *AutoAttack* with targeted attacks

To improve the performance of *AutoAttack* on the only two models where it does not achieve robust accuracy better than reported, we explore the effect of *targeted* versions of APGD and FAB, namely APGD_T and FAB_T. For APGD_T, we adapt our DLR loss to induce misclassification into a target class t by

$$\text{Targeted-DLR}(x, y) = -\frac{z_y - z_t}{z_{\pi_1} - (z_{\pi_3} + z_{\pi_4})/2}, \quad (7)$$

where we keep the notation of Sec. 4. Thus, we preserve both the shift and scaling invariance of DLR loss, while aiming at getting $z_t > z_y$, and modify the denominator in (6) to ensure that the loss is not constant. Moreover, as proposed in (Croce & Hein, 2019), FAB_T considers only the linearization of the decision boundary between the target and the correct class, instead of all the $K-1$ possible hyperplanes as in the untargeted attack. We name *AutoAttack*+ the extension of *AutoAttack* including APGD_T and FAB_T (all the hyperparameters of the targeted versions are unchanged compared to APGD and FAB).

We test *AutoAttack*+ (the details can be found in the supplement) on the deterministic models from Sec. 6 and show the results in Table 3 (see also the Appendix). APGD_T and FAB_T reduce the robust accuracy given by *AutoAttack* of 0.91% for the model of (Alayrac et al., 2019), so that it is better than the value reported in the original paper, and of 0.25% for the model of (Qin et al., 2019), getting only 0.01% far from what reported. However, let us recall that the computational budget of our evaluation with *AutoAttack*+ is still significantly lower than that granted to the attacks in (Alayrac et al., 2019) and (Qin et al., 2019). While *AutoAttack*+ improves the robust accuracy of *AutoAttack* (with roughly twice the computational cost) typically by $\leq 0.3\%$, there are a few cases (7 out of 49) where the targeted attacks lead to significantly (between 1% and 4.2%) lower values, although for models which have no SOTA robustness. Finally, it should be noted that the targeted version of APGD on the DLR loss is at least on CIFAR-10 already very competitive, outperforming even FAB.

6.2. Analysis of SOTA of adversarial defenses

While the main goal of the evaluation is to show the effectiveness of *AutoAttack*, at the same time it provides an assessment of the SOTA of adversarial defenses. The most robust defenses rely on variations or fine-tuning of adversarial training introduced in (Madry et al., 2018). One step forward has been made by methods which use additional data for training, like (Carmon et al., 2019) and (Alayrac et al., 2019). Moreover, several defenses which claim SOTA robustness turn out to be significantly less robust than (Madry et al., 2018). Interestingly, the most (empirically) resistant model on MNIST is one trained for obtaining provable certificates on the exact robust accuracy, and comes with a verified lower bound on it of 93.32% (Zhang et al., 2020).

While this paper contains up to our knowledge the largest independent evaluation of current adversarial defenses, this is by no means an exhaustive survey. Several authors did not reply to our request or were not able to provide models (or at least code). We thank all the authors who helped us in this evaluation. We hope that *AutoAttack* will contribute to a faster development of adversarial defenses and recommend it as part of a standard evaluation pipeline.

Table 3. **Robustness evaluation of adversarial defenses by *AutoAttack*+.** We report the robust accuracy attained by *AutoAttack* (AA), the *targeted* attacks APGD_T and FAB_T, and the pointwise worst-case combining them (i.e. AA+). Moreover, we show the robust accuracy reported in the original papers, and the reduction of it due to *AutoAttack*+.

#	paper	clean	AA	APGD_T	FAB_T	AA+	report.	reduct.
CIFAR-10 - l_∞ - $\epsilon = 8/255$								
1	(Carmon et al., 2019)	89.69	59.65	59.54	60.16	59.50	62.5	-3.00
2	(Alayrac et al., 2019)	86.46	56.92	56.27	57.06	56.01	56.30	-0.29
3	(Hendrycks et al., 2019)	87.11	54.99	54.94	55.25	54.86	57.4	-2.54
4	(Rice et al., 2020)	85.34	53.60	53.43	53.83	53.35	58	-4.65
5	(Qin et al., 2019)	86.28	53.07	52.85	53.28	52.82	52.81	0.01
6	(Engstrom et al., 2019)	87.03	49.58	49.32	49.81	49.20	53.29	-4.09
7	(Kumari et al., 2019)	87.80	49.40	49.15	49.56	49.06	53.04	-3.98
8	(Mao et al., 2019)	86.21	47.66	47.44	47.92	47.34	50.03	-2.69
9	(Zhang et al., 2019a)	87.20	45.06	44.85	45.39	44.78	47.98	-3.20
10	(Madry et al., 2018)	87.14	44.29	44.28	44.70	44.01	47.04	-3.03
11	(Pang et al., 2020)	80.89	43.78	43.50	44.11	43.44	55.0	-11.56
12	(Wong et al., 2020)	83.34	43.38	43.22	43.72	43.18	46.06	-2.88
13	(Ding et al., 2020)	84.36	45.57	41.74	42.37	41.41	47.18	-5.77
14	(Shafahi et al., 2019)	86.11	41.58	41.64	43.46	41.37	46.19	-4.82
15	(Moosavi-Dezfooli et al., 2019)	83.11	38.67	38.50	38.97	38.46	41.4	-2.94
16	(Zhang & Wang, 2019)	89.98	38.78	37.29	38.48	36.42	60.6	-24.18
17	(Zhang & Xu, 2020)	90.25	38.57	37.54	38.99	36.20	68.7	-32.50
18	(Jang et al., 2019)	78.91	35.09	34.96	35.48	34.88	37.40	-2.52
19	(Kim & Wang, 2020)	91.51	36.10	35.93	35.39	33.96	57.23	-23.27
20	(Moosavi-Dezfooli et al., 2019)	80.41	33.76	33.70	34.08	33.66	36.3	-2.64
21	(Wang & Zhang, 2019)	92.80	30.96	33.61	31.19	28.92	58.6	-29.68
22	(Wang & Zhang, 2019)	92.82	29.07	29.73	29.10	26.53	66.9	-40.37
23	(Mustafa et al., 2019)	89.16	0.55	1.13	0.74	0.20	32.32	-32.12
24	(Chan et al., 2020)	93.79	0.18	1.42	58.72	0.11	15.5	-15.39
25	(Pang et al., 2020)	93.52	0.00	0.00	0.00	0.00	31.4	-31.40
CIFAR-10 - l_∞ - $\epsilon = 0.031$								
1	(Zhang et al., 2019b)	84.92	53.18	53.10	53.46	53.04	56.43	-3.39
2	(Atzmon et al., 2019)	81.30	40.61	41.16	40.71	40.13	43.17	-3.04
3	(Xiao et al., 2020)	79.28	17.99	32.34	79.19	17.99	52.4	-34.41
CIFAR-100 - l_∞ - $\epsilon = 8/255$								
1	(Hendrycks et al., 2019)	59.23	28.63	28.48	28.75	28.40	33.5	-5.10
2	(Rice et al., 2020)	53.83	19.02	18.98	19.29	18.91	28.1	-9.19
MNIST - l_∞ - $\epsilon = 0.3$								
1	(Zhang et al., 2020)	98.38	93.95	94.88	96.84	93.95	96.38	-2.43
2	(Gowal et al., 2019a)	98.34	92.75	93.93	97.03	92.75	93.88	-1.13
3	(Zhang et al., 2019b)	99.48	92.76	93.62	94.68	92.74	95.60	-2.86
4	(Ding et al., 2020)	98.95	91.32	94.62	95.37	91.32	92.59	-1.27
5	(Atzmon et al., 2019)	99.35	90.85	94.16	95.25	90.85	97.35	-6.50
6	(Madry et al., 2018)	98.53	88.43	90.57	92.61	88.43	89.62	-1.19
7	(Jang et al., 2019)	98.47	87.99	93.56	94.73	87.98	94.61	-6.63
8	(Wong et al., 2020)	98.50	82.88	86.34	88.28	82.87	88.77	-5.90
9	(Taghanaki et al., 2019)	98.86	0.00	0.00	0.00	0.00	64.25	-64.25
ImageNet - l_∞ - $\epsilon = 4/255$								
1	(Engstrom et al., 2019)	63.4	28.0	27.7	28.4	27.6	33.38	-5.78
CIFAR-10 - l_2 - $\epsilon = 0.5$								
1	(Augustin et al., 2020)	91.08	73.27	72.91	73.20	72.89	73.27	-0.38
2	(Engstrom et al., 2019)	90.83	69.29	69.24	69.48	69.24	70.11	-0.87
3	(Rice et al., 2020)	88.67	67.76	67.68	67.96	67.68	71.6	-3.92
4	(Rony et al., 2019)	89.05	66.49	66.44	66.74	66.44	67.6	-1.16
5	(Ding et al., 2020)	88.02	66.13	66.09	66.34	66.09	66.18	-0.09
ImageNet - l_2 - $\epsilon = 3$								
1	(Engstrom et al., 2019)	55.3	28.6	28.3	28.5	28.3	35.09	-6.79

Acknowledgements

We are very grateful to Alvin Chan, Chengzhi Mao, Seyed-Mohsen Moosavi-Dezfooli, Chongli Qin, Saeid Asgari Taghanaki, Bao Wang and Zhi Xu for providing code, models and answering questions on their papers. We also thank Maksym Andriushchenko for insightful discussions about this work. 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

Alayrac, J.-B., Uesato, J., Huang, P.-S., Fawzi, A., Stanforth, R., and Kohli, P. Are labels required for improving adversarial robustness? In *NeurIPS*, 2019.

Andriushchenko, M., Croce, F., Flammarion, N., and Hein, M. Square attack: a query-efficient black-box adversarial attack via random search. arXiv preprint arXiv:1912.00049, 2019.

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

Atzmon, M., Haim, N., Yariv, L., Israelov, O., Maron, H., and Lipman, Y. Controlling neural level sets. In *NeurIPS*, 2019.

Augustin, M., Meinke, A., and Hein, M. Adversarial robustness on in-and out-distribution improves explainability. *arXiv preprint arXiv:2003.09461*, 2020.

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.

Carlini, N., Athalye, A., Papernot, N., Brendel, W., Rauber, J., Tsipras, D., Goodfellow, I., Madry, A., and Kurakin, A. On evaluating adversarial robustness. version from May 14, 2019, 2019. URL https://github.com/evaluating-adversarial-robustness/adv-eval-paper.

Carmon, Y., Raghunathan, A., Schmidt, L., Duchi, J. C., and Liang, P. S. Unlabeled data improves adversarial robustness. In *NeurIPS*, pp. 11190–11201, 2019.

Chan, A., Tay, Y., Ong, Y. S., and Fu, J. Jacobian adversarially regularized networks for robustness. In *ICLR*, 2020.

Cohen, J. M., Rosenfeld, E., and Kolter, J. Z. Certified adversarial robustness via randomized smoothing. In *NeurIPS*, 2019.

Croce, F. and Hein, M. Minimally distorted adversarial examples with a fast adaptive boundary attack. preprint, arXiv:1907.02044, 2019.

Croce, F., Andriushchenko, M., and Hein, M. Provable robustness of relu networks via maximization of linear regions. In *AISTATS*, 2019a.

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)*, 2019b.

Ding, G. W., Wang, L., and Jin, X. AdverTorch v0.1: An adversarial robustness toolbox based on pytorch. arXiv preprint arXiv:1902.07623, 2019.

Ding, G. W., Sharma, Y., Lui, K. Y. C., and Huang, R. Mma training: Direct input space margin maximization through adversarial training. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=HkeryxBtPB.

Engstrom, L., Ilyas, A., Santurkar, S., and Tsipras, D. Robustness (python library), 2019. URL https://github.com/MadryLab/robustness.

Goodfellow, I. J., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples. In *ICLR*, 2015.

Gowal, S., Dvijotham, K., Stanforth, R., Bunel, R., Qin, C., Uesato, J., Arandjelovic, R., Mann, T. A., and Kohli, P. On the effectiveness of interval bound propagation for training verifiably robust models. In *ICCV*, 2019a.

Gowal, S., Uesato, J., Qin, C., Huang, P.-S., Mann, T., and Kohli, P. An alternative surrogate loss for pgd-based adversarial testing. 2019b.

Grathwohl, W., Wang, K.-C., Jacobsen, J.-H., Duvenaud, D., Norouzi, M., and Swersky, K. Your classifier is secretly an energy based model and you should treat it like one. In *ICLR*, 2020.

Hendrycks, D., Lee, K., and Mazeika, M. Using pre-training can improve model robustness and uncertainty. In *ICML*, pp. 2712–2721, 2019.

Jang, Y., Zhao, T., Hong, S., and Lee, H. Adversarial defense via learning to generate diverse attacks. In *ICCV*, 2019.

Kim, J. and Wang, X. Sensible adversarial learning, 2020. URL https://openreview.net/forum?id=rJlf_RVKwr.

Kumari, N., Singh, M., Sinha, A., Machiraju, H., Krishnamurthy, B., and Balasubramanian, V. N. Harnessing the vulnerability of latent layers in adversarially trained models. In *IJCAI*, pp. 2779–2785, 7 2019.

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

Lecuyer, M., Atlidakis, V., Geambasu, R., Hsu, D., and Jana, S. Certified robustness to adversarial examples with differential privacy. In *IEEE Symposium on Security and Privacy (SP)*, 2019.

Li, B., Chen, C., Wang, W., and Carin, L. Certified adversarial robustness with additive noise. In *NeurIPS*, 2019.

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

Mao, C., Zhong, Z., Yang, J., Vondrick, C., and Ray, B. Metric learning for adversarial robustness. In *NeurIPS*, pp. 478–489, 2019.

Mirman, M., Gehr, T., and Vechev, M. Differentiable abstract interpretation for provably robust neural networks. In *ICML*, 2018.

Moosavi-Dezfooli, S.-M., Fawzi, A., Uesato, J., and Frossard, P. Robustness via curvature regularization, and vice versa. In *CVPR*, 2019.

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.

Mustafa, A., Khan, S., Hayat, M., Goecke, R., Shen, J., and Shao, L. Adversarial defense by restricting the hidden space of deep neural networks. In *ICCV*, 2019.

Pang, T., Xu, K., Du, C., Chen, N., and Zhu, J. Improving adversarial robustness via promoting ensemble diversity. In *ICML*, 2019.

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

Qin, C., Martens, J., Gowal, S., Krishnan, D., Dvijotham, K., Fawzi, A., De, S., Stanforth, R., and Kohli, P. Adversarial robustness through local linearization. In *NeurIPS*, 2019.

Rice, L., Wong, E., and Kolter, J. Z. Overfitting in adversarially robust deep learning. In *ICML*, 2020.

Rony, J., Hafemann, L. G., Oliveira, L. S., Ayed, I. B., Sabourin, R., and Granger, E. Decoupling direction and norm for efficient gradient-based l2 adversarial attacks and defenses. In *CVPR*, 2019.

Shafahi, A., Najibi, M., Ghiasi, M. A., Xu, Z., Dickerson, J., Studer, C., Davis, L. S., Taylor, G., and Goldstein, T. Adversarial training for free! In *NeurIPS*, pp. 3353–3364, 2019.

Song, C., He, K., Wang, L., and Hopcroft, J. E. Improving the generalization of adversarial training with domain adaptation. In *ICLR*, 2019.

Taghanaki, S. A., Abhishek, K., Azizi, S., and Hamarneh, G. A kernelized manifold mapping to diminish the effect of adversarial perturbations. In *CVPR*, 2019.

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

Wang, B., Shi, Z., and Osher, S. Resnets ensemble via the feynman-kac formalism to improve natural and robust accuracies. In *NeurIPS*, 2019.

Wang, J. and Zhang, H. Bilateral adversarial training: Towards fast training of more robust models against adversarial attacks. In *ICCV*, 2019.

Wong, E., Schmidt, F., Metzen, J. H., and Kolter, J. Z. Scaling provable adversarial defenses. In *NeurIPS*, 2018.

Wong, E., Rice, L., and Kolter, J. Z. Fast is better than free: Revisiting adversarial training. In *ICLR*, 2020.

Xiao, C., Zhong, P., and Zheng, C. Enhancing adversarial defense by k-winners-take-all. In *ICLR*, 2020.

Yang, Y., Zhang, G., Katabi, D., and Xu, Z. ME-net: Towards effective adversarial robustness with matrix estimation. In *ICML*, 2019.

Zhang, D., Zhang, T., Lu, Y., Zhu, Z., and Dong, B. You only propagate once: Accelerating adversarial training via maximal principle. In *NeurIPS*, pp. 227–238, 2019a.

Zhang, H. and Wang, J. Defense against adversarial attacks using feature scattering-based adversarial training. In *NeurIPS*, pp. 1829–1839, 2019.

Zhang, H. and Xu, W. Adversarial interpolation training: A simple approach for improving model robustness, 2020. URL https://openreview.net/forum?id=Syejj0NYvr.

Zhang, H., Yu, Y., Jiao, J., Xing, E. P., Ghaoui, L. E., and Jordan, M. I. Theoretically principled trade-off between robustness and accuracy. In *ICML*, 2019b.

Zhang, H., Chen, H., Xiao, C., Gowal, S., Stanforth, R., Li, B., Boning, D., and Hsieh, C.-J. Towards stable and efficient training of verifiably robust neural networks. In *ICLR*, 2020.