

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

Francesco Croce¹ Matthias Hein¹

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., 2019) 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, 2020) and the black-box Square Attack

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

(Andriushchenko et al., 2020). 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 $g : \mathcal{D} \subseteq \mathbb{R}^d \rightarrow \mathbb{R}^K$ 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 $\{z \in \mathcal{D} \mid d(x_{\text{orig}}, z) \leq \epsilon\}$. 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

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

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. $d(x, z) := \|x - z\|_p$, 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 $f : \mathbb{R}^d \rightarrow \mathbb{R}$, $\mathcal{S} \subset \mathbb{R}^d$ and the problem $\max_{x \in \mathcal{S}} f(x)$, the iterations of PGD are defined for $k = 1, \dots, N_{\text{iter}}$ as $x^{(k+1)} = P_{\mathcal{S}}(x^{(k)} + \eta^{(k)} \nabla f(x^{(k)}))$, 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

Algorithm 1 APGD

```

1: Input:  $f, \mathcal{S}, x^{(0)}, \eta, N_{\text{iter}}, W = \{w_0, \dots, w_n\}$ 
2: Output:  $x_{\text{max}}, f_{\text{max}}$ 
3:  $x^{(1)} \leftarrow P_{\mathcal{S}}(x^{(0)} + \eta \nabla f(x^{(0)}))$ 
4:  $f_{\text{max}} \leftarrow \max\{f(x^{(0)}), f(x^{(1)})\}$ 
5:  $x_{\text{max}} \leftarrow x^{(0)}$  if  $f_{\text{max}} \equiv f(x^{(0)})$  else  $x_{\text{max}} \leftarrow x^{(1)}$ 
6: for  $k = 1$  to  $N_{\text{iter}} - 1$  do
7:    $z^{(k+1)} \leftarrow P_{\mathcal{S}}(x^{(k)} + \eta \nabla f(x^{(k)}))$ 
8:    $x^{(k+1)} \leftarrow P_{\mathcal{S}}(x^{(k)} + \alpha(z^{(k+1)} - x^{(k)})$ 
      $\quad \quad \quad + (1 - \alpha)(x^{(k)} - x^{(k-1)}))$ 
9:   if  $f(x^{(k+1)}) > f_{\text{max}}$  then
10:     $x_{\text{max}} \leftarrow x^{(k+1)}$  and  $f_{\text{max}} \leftarrow f(x^{(k+1)})$ 
11:   end if
12:   if  $k \in W$  then
13:     if Condition 1 or Condition 2 then
14:        $\eta \leftarrow \eta/2$  and  $x^{(k+1)} \leftarrow x_{\text{max}}$ 
15:     end if
16:   end if
17: end for

```

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).

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

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

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. $\eta^{(w_{j-1})} \equiv \eta^{(w_j)} \text{ and } f_{\max}^{(w_{j-1})} \equiv f_{\max}^{(w_j)},$

where $f_{\max}^{(k)}$ 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 $x^{(w_j+1)} := x_{\max}$, 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 Tables 9, 10 and 11 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

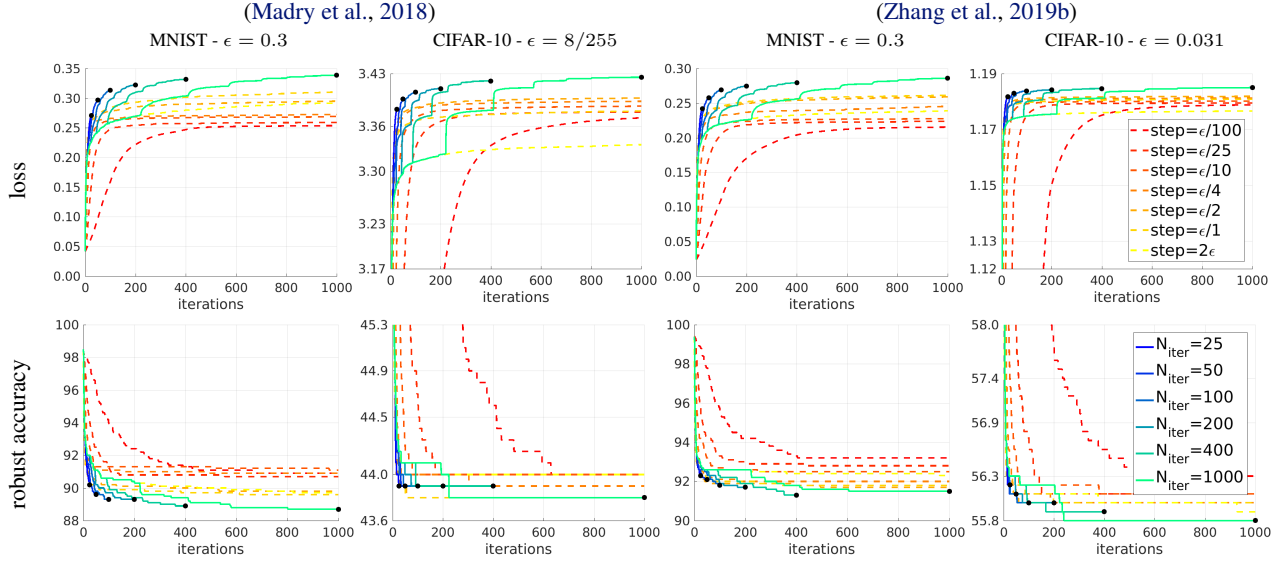


Figure 1. PGD with Momentum vs APGD: best cross-entropy loss (top) and robust accuracy (bottom) obtained so far as function of iterations for the models of (Madry et al., 2018) and TRADES (Zhang et al., 2019b) for PGD with a momentum term ($\alpha = 0.75$, as used in APGD) (dashed lines) with different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

CE loss) 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 Sec. A.1 of the Appendix.

4. An alternative loss

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

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

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

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

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

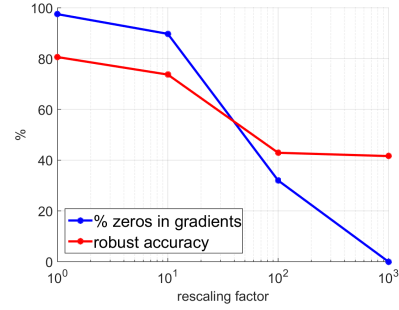


Figure 2. Percentage of zeros in the gradients and robust accuracy, computed by PGD on the CE loss, of the classifiers g/α , where g is the CIFAR-10 model of (Atzmon et al., 2019) and α a rescaling factor. The performance of PGD depends on the scale of the logits.

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 Sec. D).

The CW loss (Carlini & Wagner, 2017a) defined as

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

has in contrast to the CE loss a direct interpretation in terms of the decision of the classifier. 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 and thus has the same degrees of freedom as the decision of the classifier:

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

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 $\arg\max_i z_i \neq y$) 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. Furthermore, we adapt our DLR loss to induce misclassification into a target class t by

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

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**.

4.2. APGD versus PGD on different losses

We compare 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 C) optimizing the CE, CW and DLR loss (the complete results are reported in Tables 9, 10, 11 in Sec. C). 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). APGD outperforms the best among the 6 versions of PGD on 32 of the 43 models with CE, 37/43 with CW and 35/43 with DLR, and the models where APGD is worse are mainly the ones where the extreme step size 2ϵ is optimal as the defenses lead to gradient masking/obfuscation (further details in Sec. C.1). The version of standard PGD achieving most often the lowest robust accuracy is for all three losses PGD with Momentum and step size $\epsilon/4$, with 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%. Comparing CW and DLR loss per model (over all PGD versions and APGD) the CW loss is up to 21% worse than the DLR loss, while it is never better by more than 5% and thus the DLR-loss is more stable. In total these experiments show: i) APGD outperforms PGD/PGD with Momentum consistently regardless of the employed loss, ii) our DLR loss improves upon the CE loss and is comparable to the CW loss, but with less severe failure cases.

We run a similar comparison for the models trained to be robust wrt l_2 and report the results in Tables 12, 13 and 14. APGD again yields most often the best results for all the losses. The difference of the losses is for l_2 marginal.

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 (Crocé & Hein, 2020) and Square Attack (Andriushchenko et al., 2020), to form the ensemble *AutoAttack*, which is automatic in the sense that it does not require to specify any free parameters¹.

Since we want our protocol to be effective, computationally affordable and general, we select the following variants of the attacks to compose *AutoAttack*: APGD_{CE} without random restarts, $\text{APGD}_{\text{DLR}}^T$, i.e. on the Targeted-DLR loss (7), with 9 target classes, the targeted version of FAB, namely FAB^T , with 9 target classes and Square Attack with one run of 5000 queries. We use 100 iterations for each run of the white-box attacks. 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 been used, on average, in the evaluation of the defenses considered.

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 (Crocé & Hein, 2020). 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., 2020). Both

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

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.

The hyperparameters of all attacks in *AutoAttack* are fixed for all experiments across datasets, models and norms (see Sec. B of the Appendix). 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.

5.1. Untargeted vs targeted attacks

We here discuss why we choose the targeted versions of APGD_{DLR} and FAB, considering both the scalability and the effectiveness of the attacks. Note that we keep the untargeted formulation of the CE loss, as it is widely used and achieves the best results for randomized defenses (Table 3).

FAB: The untargeted version of FAB requires to compute at each iteration the Jacobian matrix of the classifier which has size scaling linearly with the number of classes K . While this is feasible for datasets with small K (e.g. MNIST, CIFAR-10), it becomes both computationally and memory-wise prohibitive when many classes are given, as in the case of CIFAR-100 and ImageNet. As a solution we propose to use the targeted version of FAB, namely FAB^T, as presented in (Croce & Hein, 2020), which considers only the linearization of the decision boundary between the target and the correct class, instead of all $K - 1$ possible hyperplanes as for untargeted attacks, and thus by fixing additionally the number of target classes its computational complexity and memory requirements is independent of K .

Experiments: In Table 1 we compare the untargeted attacks, APGD_{DLR} and FAB, to their respective targeted versions in terms of the robust accuracy achieved on classifiers trained with recently proposed adversarial defenses (see below for details), for different threat models. We use the untargeted methods with 5 random restarts, the targeted ones with 9 target classes, those attaining the 9 highest scores at the original point (excluding the correct one). One can see that on CIFAR-10, CIFAR-100 and ImageNet in almost all the cases (35/36 for APGD_{DLR}, 29/32 for FAB) the targeted attacks provide lower (better) results, sometimes with a large gap especially for APGD_{DLR}^T. Although on MNIST the opposite situation occurs, we show in the next section that Square Attack is a stronger adversary than both APGD_{DLR}^T and FAB^T for this dataset, and thus we favor the targeted attacks when selecting our ensemble in *AutoAttack*.

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 Sec. C) and then for randomized ones, i.e. classifiers with a stochastic component. *AutoAttack* improves almost all evaluations, showing that its good performance generalizes well across datasets, models and threat models with the same hyperparameters.

Deterministic defenses: In Tables 2 and 4 (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 original 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 (Sec. C). 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}^T, FAB^T 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 2 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 achieving a robust accuracy lower than reported.

Notably, in all but one case *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* yields also significantly lower values 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 case where it does not reduce the reported robust accuracy it is only 0.03% far from it, and this result has been obtained with a variant of PGD with 180 restarts and 200 iterations (see (Qin et al., 2019)), which is way more expensive than our evaluation.

Table 1. **Comparison untargeted vs targeted attacks.** We report clean test accuracy and robust accuracy achieved by APGD_{DLR}, APGD_{DLR}^T, FAB and FAB^T. Moreover, we show the difference between the results of the targeted and untargeted attacks, and boldface it when is negative (that is the targeted attack is stronger). FAB does not scale to CIFAR-100/ImageNet due to the large number of classes.

#	paper	clean	APGD _{DLR}	APGD _{DLR} ^T	diff.	FAB	FAB ^T	diff.
CIFAR-10 - $l_\infty - \epsilon = 8/255$								
1	(Carmon et al., 2019)	89.69	60.64	59.54	-1.10	60.62	60.12	-0.50
2	(Alayrac et al., 2019)	86.46	62.03	56.27	-5.76	58.20	56.81	-1.39
3	(Hendrycks et al., 2019)	87.11	56.96	54.94	-2.02	55.40	55.27	-0.13
4	(Rice et al., 2020)	85.34	55.72	53.43	-2.29	54.13	53.83	-0.30
5	(Qin et al., 2019)	86.28	55.46	52.85	-2.61	53.77	53.28	-0.49
6	(Engstrom et al., 2019)	87.03	52.65	49.32	-3.33	50.37	49.81	-0.56
7	(Kumari et al., 2019)	87.80	51.68	49.15	-2.53	49.87	49.54	-0.33
8	(Mao et al., 2019)	86.21	50.33	47.44	-2.89	48.32	47.91	-0.41
9	(Zhang et al., 2019a)	87.20	47.33	44.85	-2.48	45.66	45.39	-0.27
10	(Madry et al., 2018)	87.14	46.03	44.28	-1.75	45.41	44.75	-0.66
11	(Pang et al., 2020)	80.89	44.56	43.50	-1.06	44.47	44.06	-0.41
12	(Wong et al., 2020)	83.34	46.64	43.22	-3.42	44.05	43.74	-0.31
13	(Shafahi et al., 2019)	86.11	44.56	41.64	-2.92	42.90	43.44	0.54
14	(Ding et al., 2020)	84.36	50.26	41.74	-8.52	47.18	42.47	-4.71
15	(Moosavi-Dezfooli et al., 2019)	83.11	40.29	38.50	-1.79	39.04	38.97	-0.07
16	(Zhang & Wang, 2019)	89.98	48.96	37.29	-11.67	40.84	38.48	-2.36
17	(Zhang & Xu, 2020)	90.25	49.40	37.54	-11.86	40.36	38.99	-1.37
18	(Jang et al., 2019)	78.91	37.01	34.96	-2.05	35.54	35.50	-0.04
19	(Kim & Wang, 2020)	91.51	48.41	35.93	-12.48	38.88	35.41	-3.47
20	(Moosavi-Dezfooli et al., 2019)	80.41	35.47	33.70	-1.77	34.08	34.08	0.00
21	(Wang & Zhang, 2019)	92.80	40.33	33.61	-6.72	33.51	31.19	-2.32
22	(Wang & Zhang, 2019)	92.82	36.58	29.73	-6.85	31.82	29.10	-2.72
23	(Mustafa et al., 2019)	89.16	4.54	1.13	-3.41	1.12	0.71	-0.41
24	(Pang et al., 2020)	93.52	0.49	0.00	-0.49	0.03	0.00	-0.03
CIFAR-10 - $l_\infty - \epsilon = 0.031$								
1	(Zhang et al., 2019b)	84.92	54.04	53.10	-0.94	53.79	53.45	-0.34
2	(Atzmon et al., 2019)	81.30	44.50	41.16	-3.34	40.92	40.73	-0.19
3	(Xiao et al., 2020)	79.28	31.27	32.34	1.07	79.28	79.28	0.00
CIFAR-100 - $l_\infty - \epsilon = 8/255$								
1	(Hendrycks et al., 2019)	59.23	31.66	28.48	-3.18	-	28.74	-
2	(Rice et al., 2020)	53.83	20.25	18.98	-1.27	-	19.24	-
MNIST - $l_\infty - \epsilon = 0.3$								
1	(Zhang et al., 2020)	98.38	94.77	94.88	0.11	95.60	96.84	1.24
2	(Gowal et al., 2019)	98.34	93.84	93.93	0.09	94.72	97.03	2.31
3	(Zhang et al., 2019b)	99.48	93.96	93.58	-0.38	94.12	94.62	0.50
4	(Ding et al., 2020)	98.95	94.03	94.62	0.59	94.33	95.37	1.04
5	(Atzmon et al., 2019)	99.35	94.54	94.16	-0.38	94.12	95.26	1.14
6	(Madry et al., 2018)	98.53	89.75	90.57	0.82	91.75	93.69	1.94
7	(Jang et al., 2019)	98.47	92.15	93.56	1.41	93.24	94.74	1.50
8	(Wong et al., 2020)	98.50	85.39	86.34	0.95	87.30	88.28	0.98
9	(Taghanaki et al., 2019)	98.86	0.00	0.00	0.00	0.02	0.01	-0.01
ImageNet - $l_\infty - \epsilon = 4/255$								
1	(Engstrom et al., 2019)	63.4	32.0	27.7	-4.3	-	28.4	-
CIFAR-10 - $l_2 - \epsilon = 0.5$								
1	(Augustin et al., 2020)	91.08	74.94	72.91	-2.03	74.13	73.18	-0.95
2	(Engstrom et al., 2019)	90.83	70.20	69.24	-0.96	69.54	69.46	-0.08
3	(Rice et al., 2020)	88.67	68.95	67.68	-1.27	68.03	67.97	-0.06
4	(Rony et al., 2019)	89.05	67.02	66.44	-0.58	66.81	66.74	-0.07
5	(Ding et al., 2020)	88.02	66.53	66.09	-0.44	66.43	66.33	-0.10
ImageNet - $l_2 - \epsilon = 3$								
1	(Engstrom et al., 2019)	55.3	30.9	28.3	-2.6	-	28.5	-

Table 2. **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} ^T	FAB ^T	Square	AA	reported	reduct.
CIFAR-10 - l_∞ - $\epsilon = 8/255$									
1	(Carmon et al., 2019)	89.69	61.74	59.54	60.12	66.63	59.53	62.5	-2.97
2	(Alayrac et al., 2019)	86.46	60.17	56.27	56.81	66.37	56.03	56.30	-0.27
3	(Hendrycks et al., 2019)	87.11	<u>57.23</u>	54.94	<u>55.27</u>	61.99	54.92	57.4	-2.48
4	(Rice et al., 2020)	85.34	<u>57.00</u>	53.43	<u>53.83</u>	61.37	53.42	58	-4.58
5	(Qin et al., 2019)	86.28	55.70	52.85	53.28	60.01	52.84	52.81	0.03
6	(Engstrom et al., 2019)	87.03	<u>51.72</u>	49.32	<u>49.81</u>	58.12	49.25	53.29	-4.04
7	(Kumari et al., 2019)	87.80	<u>51.80</u>	49.15	<u>49.54</u>	58.20	49.12	53.04	-3.92
8	(Mao et al., 2019)	86.21	49.65	47.44	<u>47.91</u>	56.98	47.41	50.03	-2.62
9	(Zhang et al., 2019a)	87.20	<u>46.15</u>	44.85	<u>45.39</u>	55.08	44.83	47.98	-3.15
10	(Madry et al., 2018)	87.14	<u>44.75</u>	44.28	<u>44.75</u>	53.10	44.04	47.04	-3.00
11	(Pang et al., 2020)	80.89	57.07	43.50	<u>44.06</u>	49.73	43.48	55.0	-11.52
12	(Wong et al., 2020)	83.34	45.90	43.22	<u>43.74</u>	53.32	43.21	46.06	-2.85
13	(Shafahi et al., 2019)	86.11	<u>43.66</u>	41.64	<u>43.44</u>	51.95	41.47	46.19	-4.72
14	(Ding et al., 2020)	84.36	50.12	41.74	<u>42.47</u>	55.53	41.44	47.18	-5.74
15	(Moosavi-Dezfooli et al., 2019)	83.11	41.72	38.50	<u>38.97</u>	47.69	38.50	41.4	-2.90
16	(Zhang & Wang, 2019)	89.98	64.42	37.29	<u>38.48</u>	59.12	36.64	60.6	-23.96
17	(Zhang & Xu, 2020)	90.25	71.40	37.54	<u>38.99</u>	66.88	36.45	68.7	-32.25
18	(Jang et al., 2019)	78.91	37.76	34.96	<u>35.50</u>	44.33	34.95	37.40	-2.45
19	(Kim & Wang, 2020)	91.51	<u>56.64</u>	<u>35.93</u>	35.41	61.30	34.22	57.23	-23.01
20	(Moosavi-Dezfooli et al., 2019)	80.41	36.65	33.70	<u>34.08</u>	43.46	33.70	36.3	-2.60
21	(Wang & Zhang, 2019)	92.80	59.09	<u>33.61</u>	31.19	64.22	29.35	58.6	-29.25
22	(Wang & Zhang, 2019)	92.82	69.62	<u>29.73</u>	<u>29.10</u>	66.77	26.93	66.9	-39.97
23	(Mustafa et al., 2019)	89.16	8.16	1.13	0.71	33.91	0.28	32.32	-32.04
24	(Chan et al., 2020)	93.79	<u>2.06</u>	0.53	58.13	71.43	0.26	15.5	-15.24
25	(Pang et al., 2020)	93.52	89.48	0.00	0.00	35.82	0.00	31.4	-31.40
CIFAR-10 - l_∞ - $\epsilon = 0.031$									
1	(Zhang et al., 2019b)	84.92	<u>55.28</u>	53.10	<u>53.45</u>	59.43	53.08	56.43	-3.35
2	(Atzmon et al., 2019)	81.30	79.67	<u>41.16</u>	40.73	47.99	40.22	43.17	-2.95
3	(Xiao et al., 2020)	79.28	<u>39.99</u>	<u>32.34</u>	<u>79.28</u>	20.44	18.50	52.4	-33.90
CIFAR-100 - l_∞ - $\epsilon = 8/255$									
1	(Hendrycks et al., 2019)	59.23	<u>33.02</u>	28.48	<u>28.74</u>	34.26	28.42	33.5	-5.08
2	(Rice et al., 2020)	53.83	<u>20.57</u>	18.98	<u>19.24</u>	<u>23.57</u>	18.95	28.1	-9.15
MNIST - l_∞ - $\epsilon = 0.3$									
1	(Zhang et al., 2020)	98.38	<u>95.32</u>	<u>94.88</u>	96.84	93.97	93.96	96.38	-2.42
2	(Gowal et al., 2019)	98.34	94.79	93.93	97.03	92.88	92.83	93.88	-1.05
3	(Zhang et al., 2019b)	99.48	<u>93.60</u>	<u>93.58</u>	<u>94.62</u>	92.97	92.81	95.60	-2.79
4	(Ding et al., 2020)	98.95	94.58	94.62	95.37	91.42	91.40	92.59	-1.19
5	(Atzmon et al., 2019)	99.35	99.10	<u>94.16</u>	<u>95.26</u>	90.86	90.85	97.35	-6.50
6	(Madry et al., 2018)	98.53	90.57	90.57	93.69	88.56	88.50	89.62	-1.12
7	(Jang et al., 2019)	98.47	<u>94.05</u>	<u>93.56</u>	94.74	88.00	87.99	94.61	-6.62
8	(Wong et al., 2020)	98.50	86.68	<u>86.34</u>	<u>88.28</u>	83.07	82.93	88.77	-5.84
9	(Taghanaki et al., 2019)	98.86	<u>30.50</u>	0.00	<u>0.01</u>	0.00	0.00	64.25	-64.25
ImageNet - l_∞ - $\epsilon = 4/255$									
1	(Engstrom et al., 2019)	63.4	<u>31.0</u>	27.7	<u>28.4</u>	46.8	27.6	33.38	-5.78
CIFAR-10 - l_2 - $\epsilon = 0.5$									
1	(Augustin et al., 2020)	91.08	74.70	72.91	<u>73.18</u>	83.10	72.91	73.27	-0.36
2	(Engstrom et al., 2019)	90.83	<u>69.62</u>	69.24	<u>69.46</u>	80.92	69.24	70.11	-0.87
3	(Rice et al., 2020)	88.67	68.58	67.68	<u>67.97</u>	79.01	67.68	71.6	-3.92
4	(Rony et al., 2019)	89.05	<u>66.59</u>	66.44	<u>66.74</u>	78.05	66.44	67.6	-1.16
5	(Ding et al., 2020)	88.02	66.21	66.09	66.33	76.99	66.09	66.18	-0.09
ImageNet - l_2 - $\epsilon = 3$									
1	(Engstrom et al., 2019)	55.3	<u>31.5</u>	28.3	<u>28.5</u>	46.6	28.3	35.09	-6.79

Table 3. 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 one 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	<i>AutoAttack</i>	report.	reduct.
CIFAR-10 - $\epsilon = 8/255$										
1	(Wang et al., 2019)	EnRN	82.39 (0.14)	<u>48.81</u>	<u>49.37</u>	-	78.61	45.56 (0.20)	51.48	-5.9
2	(Yang et al., 2019)	with AT	84.9 (0.6)	<u>30.1</u>	<u>31.9</u>	-	-	26.3 (0.85)	52.8	-26.5
3	(Yang et al., 2019)	pure	87.2 (0.3)	<u>21.5</u>	<u>24.3</u>	-	-	18.2 (0.82)	40.8	-22.6
4	(Grathwohl et al., 2020)	JEM-10	90.99 (0.03)	<u>11.69</u>	<u>15.88</u>	63.07	79.32	9.92 (0.03)	47.6	-37.7
5	(Grathwohl et al., 2020)	JEM-1	92.31 (0.04)	<u>9.15</u>	<u>13.85</u>	62.71	79.25	8.15 (0.05)	41.8	-33.6
6	(Grathwohl et al., 2020)	JEM-0	92.82 (0.05)	<u>7.19</u>	<u>12.63</u>	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)	<u>49.10</u>	<u>52.55</u>	78.87	89.32	47.97 (0.05)	72.6	-24.6
2	(Grathwohl et al., 2020)	JEM-1	92.34 (0.04)	<u>46.08</u>	<u>49.71</u>	78.93	90.17	45.49 (0.04)	67.1	-21.6
3	(Grathwohl et al., 2020)	JEM-0	92.82 (0.02)	<u>42.98</u>	<u>47.74</u>	82.92	89.52	42.55 (0.07)	50.8	-8.2

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 21/49 cases, APGD_{DLR} in 45/49, FAB^T in 39/49 and Square Attack in 17/49, but 9/9 on MNIST). APGD_{DLR} most often attains the best result for CIFAR-10, CIFAR-100 and ImageNet, Square Attack on MNIST. Also, 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 12%, compared to 89% for APGD_{CE}, 59% for FAB^T 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 (hence the decision) of the model might change across different runs for the same input. Thus we compute in Table 3 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 across 5 runs. To counter the randomness of the classifiers, for APGD we compute the direction for the update step as the average of 20 computations of the gradient at the same point (known as Expectation over Transformation (Athalye et al., 2018)) and use the untargeted losses (1 run). 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. We modify Square Attack to accept an update if it reduces the target loss on average over 20 forward passes and, as this costs more time we use only 1000 iterations. For the models from (Grathwohl et al., 2020), we attack that named JEM-0 with 5 restarts with the deterministic versions (i.e. without averaging across multiple passes of the networks), since the stochastic component has little

influence, and then reuse the same adversarial examples on JEM-1 and JEM-10 (the results of FAB confirm that it is not suitable to test randomized defenses). Table 3 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 cases larger than 25%). Thus *AutoAttack* is also suitable for the evaluation of randomized defenses.

6.1. 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 as it is quick and requires no hyperparameter tuning.

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. In *ECCV*, 2020.
- 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. In *ECCV*, 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., Raghuathan, 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. In *ICML*, 2020.
- 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 *ICLR*, 2020.
- 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*, 2019.
- 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., Goyal, 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., Goyal, S., Stanforth, R., Li, B., Boning, D., and Hsieh, C.-J. Towards stable and efficient training of verifiably robust neural networks. In *ICLR*, 2020.

A. Auto-PGD

In the main paper we compare in Sec. 3.2 the performance of Auto-PGD versus PGD with Momentum with different fixed step sizes as a function of the iterations to show that Auto-PGD adapts automatically to a given budget of iterations and to a good step size, achieving larger (better) loss and smaller (better) robust test accuracy than PGD with momentum. In Sec. A.1 we show that the same holds if we compare Auto-PGD to PGD (without momentum). Moreover, we provide in Sec. C.1 a full comparison of PGD, PGD with Momentum, both with three different step sizes, and Auto-PGD. It turns out that Auto-PGD outperforms PGD with a fixed step size showing again that the automatic adaptation of Auto-PGD works well.

A.1. Comparison of APGD to PGD

We repeat the experiment of Sec. 3.2, this time comparing the performance of APGD to that of PGD for various fixed step sizes as a function of the iterations.

The results of the comparison of APGD vs PGD (without momentum) are shown in Figure 3. PGD without momentum achieves significantly smaller loss values on MNIST and also worse robust accuracy than PGD with Momentum and in particular APGD. Apart from this the behavior of PGD with fixed step size is very similar to the one of PGD with momentum where as we see that the loss and robust accuracy plateau quickly while APGD manages to get still improvements. In general, APGD outperforms PGD in terms of both loss and robust accuracy.

B. AutoAttack: implementation details

We report all the hyperparameters used in *AutoAttack*. For APGD, we use as direction in the update step the sign of the gradient for the l_∞ -threat model and, as common in literature (see e.g. (Tsipras et al., 2019)), the normalized (wrt l_2) gradient for the l_2 -threat model. Moreover, we set the momentum coefficient to $\alpha = 0.75$, $\rho = 0.75$ (Condition 1), initial step size $\eta^{(0)} = 2\epsilon$ where ϵ is the maximum l_p -norm of the perturbations. For FAB we keep the standard hyperparameters according to the implementation of Advtorch² (Ding et al., 2019). For Square Attack we follow the original code³, using as initial value for the size of the squares $p = 0.8$. Moreover, we use the piecewise constant schedule for p as it is suggested for a query limit of 10000 without any rescaling (although we use only up to 5000 queries).

²<https://github.com/BorealisAI/advtorch>

³<https://github.com/max-andr/square-attack>

C. Experiments

Deterministic defenses: We report in Table 4 the results for the deterministic defenses wrt l_∞ omitted in Sec. 6, involving CIFAR-10 models with less common thresholds ϵ , with the same statistics as in Table 2. We add in Tables 5 and 6 (for l_∞ - and l_2 -defenses respectively) more details about the classifiers and papers considered in our evaluation: the venue where each paper appeared (we add "R" to indicate that the paper was submitted at the venue but rejected) and the source of the models, that is whether they are publicly *available*, we got them via private communication from the *authors* or we *retrained* them according to official implementations. We also indicate the architecture of the classifiers: interestingly, larger models do not imply better robustness.

The models of (Pang et al., 2019) and (Pang et al., 2020) appears twice as they are evaluated both with and without the addition of adversarial training. The models of (Wang & Zhang, 2019) are the networks named "R-MOSA-LA-4" and "R-MC-LA-4".

Randomized defenses: Similar to the deterministic defenses we report additional information which had to be omitted in Table 3. We report in Table 7 the venue where each paper appeared and the source of the models, that is if they are publicly *available* or we got them via private communication from the *authors*. In Table 8 we report the mean and standard deviation of the robust accuracy achieved by each attack (note that we have a randomized defense thus the outcome is not deterministic). As mentioned in Sec. 6, to evaluate an attack we compute 5 times the classification accuracy of the target model on the same adversarial examples. For *AutoAttack*, we form for each test point a batch with the adversarial examples among those of our different attacks which were misclassified most often in the 5 runs, and then compute the robust accuracy of this batch.

For the models of (Grathwohl et al., 2020), only plots of robust accuracy vs size of the perturbation ϵ are available in their paper. Thus the reported values are obtained by extrapolating the values in the tables from the most recent version of the paper, that is the one available at <https://arxiv.org/abs/1912.03263v2>.

C.1. Comparison PGD vs PGD with Momentum vs APGD on different losses

We add here more details about the comparison introduced in Sec. 4.2. In order to compare the performance of APGD to that of standard PGD when testing adversarial defenses, we run PGD and PGD with Momentum on the models used in Sec. 6, with the same budget of APGD, i.e. 100 iterations and 5 restarts. We use three step sizes: $\epsilon/10$, $\epsilon/4$ and 2ϵ , where ϵ is the threshold at which the robust accuracy is

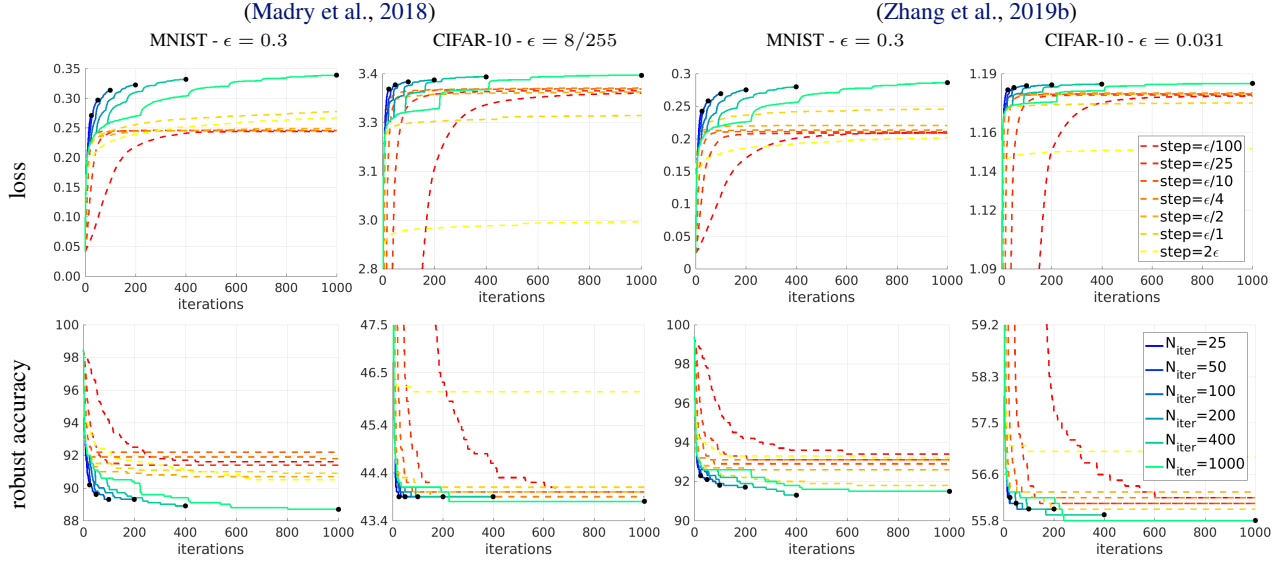


Figure 3. **PGD vs APGD**: best cross-entropy loss (top) and robust accuracy (bottom) so far found as function of iterations for the models of (Madry et al., 2018) and TRADES (Zhang et al., 2019b) for PGD (dashed lines) with different fixed step sizes (always 1000 iterations) and APGD (solid lines) with different budgets of iterations. APGD outperforms PGD for every budget of iterations in robust accuracy.

Table 4. **Robustness evaluation of additional adversarial defenses by AutoAttack**. We report clean test accuracy, the robust accuracy of the individual attacks and the combined one of AutoAttack (AA column). We also provide the robust accuracy reported in the original papers and compute the difference to that of AutoAttack. If negative (in red) AutoAttack provides lower (better) robust accuracy.

#	paper	clean	APGD _{CE}	APGD _{DLR} ^T	FAB ^T	Square	AA	reported	reduct.
CIFAR-10 - $l_\infty - \epsilon = 4/255$									
1	(Song et al., 2019)	84.81	<u>57.43</u>	<u>56.51</u>	<u>56.86</u>	64.67	56.51	58.1	-1.59
CIFAR-10 - $l_\infty - \epsilon = 0.02$									
1	(Pang et al., 2019)	91.22	<u>5.02</u>	<u>1.74</u>	<u>2.45</u>	<u>29.52</u>	1.02	34.0	-32.98
2	(Pang et al., 2019)	93.44	<u>0.18</u>	<u>0.01</u>	<u>0.06</u>	<u>88.55</u>	0.00	30.4	-30.40

computed. We repeat the experiment for the CE (Tables 9 and 12), CW (Tables 10 and 13) and DLR loss (Tables 11 and 14).

We see that APGD achieves with all losses the lowest (lower is better) robust accuracy in most of the cases (36 out of 49 with CE loss, 42/49 with CW loss, 39/49 with the DLR loss). Moreover, while on average PGD with Momentum and step size $\epsilon/4$ is slightly better than the other versions of PGD, the best version of PGD varies with the classifier, and in particular there is large variance among the results obtained by different step sizes. This emphasizes again why a step size free method is essential for a reliable robustness evaluation which is as much as possible independent of the user.

For the attacks on the CW and DLR loss, the only cases where PGD is more than 1% better than APGD consists of cases where the very large step size 2ϵ outperforms all other step sizes. Note that the stepsize 2ϵ works significantly worse than the smaller step sizes for most of the

other models. It is likely that these defenses modify the loss landscape to make it unsuitable for standard gradient-based optimization, and an *informed random search* (as PGD with such a large step size can be seen) works better. The same happens with the CE loss, where PGD yields the best results (with a non-negligible gap to APGD) also on the models of (Kim & Wang, 2020) and (Taghanaki et al., 2019). However, notice that for these classifiers optimizing the DLR loss is significantly more effective than using the CE loss. Finally, in all these cases AutoAttack achieves significantly lower robust accuracy than any version of PGD as FAB attack is less affected by this kind of gradient obfuscation.

D. Potential failure cases of APGD_{CE} and APGD_{DLR}

It is always important to understand in which cases some attacks do not work. The automatic step size selection in APGD prevents a lot of the failure cases of PGD with a fixed

step size. However, clearly both losses are affected when the gradient information in general is not helpful e.g. the defense of (Xiao et al., 2020) leads to a discontinuous classifier function and thus APGD_{CE} and APGD_{DLR} overestimate the robustness (even though they still get much lower robust accuracy than (Xiao et al., 2020)). However, this particular defense shows the value of *AutoAttack* as in this case the black-box Square Attack which does not rely on gradient information reveals that this defense is not robust. In fact we have run Square attack with a query limit of 10000 and 10 random restarts (instead of 5000 and 1 in the experiments) and got a robust accuracy of 8.0% (on 500 points) compared to 20.44% in our experiments.

D.1. Cross-entropy loss

As mentioned in Sec. 4, for the cross-entropy loss a problem is that the loss is not invariant to rescaling of the classifier. The finite arithmetic leads to the fact that when its gradient, in Eq. 4, is analytically $\nabla_x \text{CE}(x, y) \approx \mathbf{0}$, it becomes in practice $\nabla_x \text{CE}(x, y) = \mathbf{0}$, since typically single precision is used and only exponents roughly in $[-127, 127]$ in the exponential function can be expressed (recently (Wong et al., 2020) proposed even half precision). Thus the sign of the gradient (which in principle is not affected by the magnitude of the values) becomes zero and one does not get meaningful ascent directions (known as gradient masking).

However, we highlight that maximizing the CE loss minimizes the confidence of the classifier in the correct class. This helps to test randomized defenses, when misclassification alone is not enough for a successful attack.

D.2. Difference-of-Logits Ratio (DLR) loss

As discussed in Sec. 4.2 the DLR loss is the best performing loss on average when integrated into APGD. As it is rescaling and shift-invariant it does not suffer from the issues of the cross-entropy loss. In our large scale evaluation APGD_{DLR} is the best performing attack in the sense that it has the lowest maximal difference to best performing attack across all models we tested. However, as discussed above APGD_{DLR} can potentially perform not very well for discontinuous classifiers as seen in the evaluation of (Xiao et al., 2020).

Table 5. **Robustness evaluation of l_∞ -adversarial defenses with *AutoAttack*.** For each model, we report architecture, source, venue, clean accuracy and combined robust accuracy given by *AutoAttack* (AA column). We also provide the reported robust accuracy in the original papers and compute the difference to the one of *AutoAttack*.

#	paper	model	source	venue	clean	AA	reported	reduct.
CIFAR-10 - $\epsilon = 8/255$								
1	(Carmon et al., 2019)	WideResNet-28-10	available	NeurIPS 19	89.69	59.53	62.5	-2.97
2	(Alayrac et al., 2019)	WideResNet-106-8	available	NeurIPS 19	86.46	56.03	56.30	-0.27
3	(Hendrycks et al., 2019)	WideResNet-28-10	available	ICML 19	87.11	54.92	57.4	-2.48
4	(Rice et al., 2020)	WideResNet-34-20	available	ICML 20	85.34	53.42	58	-4.58
5	(Qin et al., 2019)	WideResNet-40-8	available	NeurIPS 19	86.28	52.84	52.81	0.03
6	(Engstrom et al., 2019)	ResNet-50	available	GitHub	87.03	49.25	53.29	-4.04
7	(Kumari et al., 2019)	WideResNet-34-10	available	IJCAI 19	87.80	49.12	53.04	-3.92
8	(Mao et al., 2019)	WideResNet-34-10	authors	NeurIPS 19	86.21	47.41	50.03	-2.62
9	(Zhang et al., 2019a)	WideResNet-34-10	retrained	NeurIPS 19	87.20	44.83	47.98	-3.15
10	(Madry et al., 2018)	WideResNet-34-10	available	ICLR 18	87.14	44.04	47.04	-3.00
11	(Pang et al., 2020)	ResNet32	available	ICLR 20	80.89	43.48	55.0	-11.52
12	(Wong et al., 2020)	ResNet18	available	ICLR 20	83.34	43.21	46.06	-2.85
13	(Shafahi et al., 2019)	WideResNet-34-10	available	NeurIPS 19	86.11	41.47	46.19	-4.72
14	(Ding et al., 2020)	WideResNet-28-4	available	ICLR 20	84.36	41.44	47.18	-5.74
15	(Moosavi-Dezfooli et al., 2019)	WideResNet-28-10	authors	CVPR 19	83.11	38.50	41.4	-2.90
16	(Zhang & Wang, 2019)	WideResNet-28-10	available	NeurIPS 19	89.98	36.64	60.6	-23.96
17	(Zhang & Xu, 2020)	WideResNet-28-10	available	ICLR 20 R	90.25	36.45	68.7	-32.25
18	(Jang et al., 2019)	ResNet20	available	ICCV 19	78.91	34.95	37.40	-2.45
19	(Kim & Wang, 2020)	WideResNet-34-10	available	ICLR 20 R	91.51	34.22	57.23	-23.01
20	(Moosavi-Dezfooli et al., 2019)	ResNet18	authors	CVPR 19	80.41	33.70	36.3	-2.60
21	(Wang & Zhang, 2019)	WideResNet-28-10	available	ICCV 19	92.80	29.35	58.6	-29.25
22	(Wang & Zhang, 2019)	WideResNet-28-10	available	ICCV 19	92.82	26.93	66.9	-39.97
23	(Mustafa et al., 2019)	ResNet110	available	ICCV 19	89.16	0.28	32.32	-32.04
24	(Chan et al., 2020)	WideResNet-34-10	retrained	ICLR 20	93.79	0.26	15.5	-15.24
25	(Pang et al., 2020)	ResNet110	available	ICLR 20	93.52	0.00	31.4	-31.40
CIFAR-10 - $l_\infty - \epsilon = 0.031$								
1	(Zhang et al., 2019b)	WideResNet-34-10	available	ICML 19	84.92	53.08	56.43	-3.35
2	(Atzmon et al., 2019)	ResNet18	available	NeurIPS 19	81.30	40.22	43.17	-2.95
3	(Xiao et al., 2020)	DenseNet121	available	ICLR 20	79.28	18.50	52.4	-33.90
CIFAR-10 - $l_\infty - \epsilon = 4/255$								
1	(Song et al., 2019)	ConvNet	available	ICLR 19	84.81	56.51	58.1	-1.59
CIFAR-10 - $l_\infty - \epsilon = 0.02$								
1	(Pang et al., 2019)	ResNet20 (x3)	available	ICML 19	91.22	1.02	34.0	-32.98
2	(Pang et al., 2019)	ResNet20 (x3)	available	ICML 19	93.44	0.00	30.4	-30.40
CIFAR-100 - $l_\infty - \epsilon = 8/255$								
1	(Hendrycks et al., 2019)	WideResNet-28-10	available	ICML 19	59.23	28.42	33.5	-5.08
2	(Rice et al., 2020)	PreActResNet-18	available	ICML 20	53.83	18.95	28.1	-9.15
MNIST - $l_\infty - \epsilon = 0.3$								
1	(Zhang et al., 2020)		available	ICLR 20	98.38	93.96	96.38	-2.42
2	(Gowal et al., 2019)		available	ICCV 19	98.34	92.83	93.88	-1.05
3	(Zhang et al., 2019b)		available	ICML 19	99.48	92.81	95.60	-2.79
4	(Ding et al., 2020)		available	ICLR 20	98.95	91.40	92.59	-1.19
5	(Atzmon et al., 2019)		available	NeurIPS 19	99.35	90.85	97.35	-6.50
6	(Madry et al., 2018)		available	ICLR 18	98.53	88.50	89.62	-1.12
7	(Jang et al., 2019)		available	ICCV 19	98.47	87.99	94.61	-6.62
8	(Wong et al., 2020)		available	ICLR 20	98.50	82.93	88.77	-5.84
9	(Taghanaki et al., 2019)		retrained	CVPR 19	98.86	0.00	64.25	-64.25
ImageNet - $l_\infty - \epsilon = 4/255$								
1	(Engstrom et al., 2019)	ResNet-50	available	GitHub	63.4	27.6	33.38	-5.78

Table 6. **Robustness evaluation of l_2 -adversarial defenses with *AutoAttack*.** For each model, we report architecture, source, venue, clean accuracy and combined robust accuracy given by *AutoAttack* (AA column). We also provide the reported robust accuracy in the original papers and compute the difference to the one of *AutoAttack*.

#	paper	model	source	venue	clean	AA	reported	reduct.
CIFAR-10 - $l_2 - \epsilon = 0.5$								
1	(Augustin et al., 2020)	ResNet-50	authors	ECCV 20	91.08	72.91	73.27	-0.36
2	(Engstrom et al., 2019)	ResNet-50	available	GitHub	90.83	69.24	70.11	-0.87
3	(Rice et al., 2020)	PreActResNet-18	available	ICML 20	88.67	67.68	71.6	-3.92
4	(Rony et al., 2019)	WideResNet-28-10	available	CVPR 19	89.05	66.44	67.6	-1.16
5	(Ding et al., 2020)	WideResNet-28-4	available	ICLR 20	88.02	66.09	66.18	-0.09
ImageNet - $l_2 - \epsilon = 3$								
1	(Engstrom et al., 2019)	ResNet-50	available	GitHub	55.3	28.3	35.09	-6.79

Table 7. **Robustness evaluation of randomized l_∞ -adversarial defenses by *AutoAttack*.** For each model, we report architecture, source, venue, clean accuracy and combined robust accuracy given by *AutoAttack* (AA column). We also provide the reported robust accuracy in the original papers and compute the difference to the one of *AutoAttack*.

#	paper	model	source	venue	clean	<i>AutoAttack</i>	report.	reduct.
CIFAR-10 - $\epsilon = 8/255$								
1	(Wang et al., 2019)	En ₅ RN	authors	NeurIPS 19	82.39 (0.14)	45.56 (0.20)	51.48	-5.9
2	(Yang et al., 2019)	with AT	authors	ICML 19	84.9 (0.6)	26.3 (0.85)	52.8	-26.5
3	(Yang et al., 2019)	pure	authors	ICML 19	87.2 (0.3)	18.2 (0.82)	40.8	-22.6
4	(Grathwohl et al., 2020)	JEM-10	available	ICLR 20	90.99 (0.03)	9.92 (0.03)	47.6	-37.7
5	(Grathwohl et al., 2020)	JEM-1	available	ICLR 20	92.31 (0.04)	8.15 (0.05)	41.8	-33.6
6	(Grathwohl et al., 2020)	JEM-0	available	ICLR 20	92.82 (0.05)	6.36 (0.06)	19.8	-13.4
CIFAR-10 - $\epsilon = 4/255$								
1	(Grathwohl et al., 2020)	JEM-10	available	ICLR 20	91.03 (0.05)	47.97 (0.05)	72.6	-24.6
2	(Grathwohl et al., 2020)	JEM-1	available	ICLR 20	92.34 (0.04)	45.49 (0.04)	67.1	-21.6
3	(Grathwohl et al., 2020)	JEM-0	available	ICLR 20	92.82 (0.02)	42.55 (0.07)	50.8	-8.2

Table 8. **Robustness evaluation of randomized l_∞ -adversarial defenses by *AutoAttack*.** We report mean and standard deviation over 5 runs of evaluation of the robust accuracy achieved by the adversarial examples crafted by the different attacks, together with their combination in *AutoAttack*.

#	paper	model	APGD _{CE}	APGD _{DLR}	FAB	Square	<i>AutoAttack</i>
CIFAR-10 - $\epsilon = 8/255$							
1	(Wang et al., 2019)	En ₅ RN	48.81 (0.16)	49.37 (0.17)	-	78.61 (0.19)	45.56 (0.20)
2	(Yang et al., 2019)	with AT	30.1 (0.5)	31.9 (0.53)	-	-	26.3 (0.85)
3	(Yang et al., 2019)	pure	21.5 (0.6)	24.3 (0.98)	-	-	18.2 (0.82)
4	(Grathwohl et al., 2020)	JEM-10	11.69 (0.04)	15.88 (0.06)	63.07 (0.21)	79.32 (0.04)	9.92 (0.03)
5	(Grathwohl et al., 2020)	JEM-1	9.15 (0.02)	13.85 (0.05)	62.71 (0.17)	79.25 (0.14)	8.15 (0.05)
6	(Grathwohl et al., 2020)	JEM-0	7.19 (0.06)	12.63 (0.04)	66.48 (0.34)	73.12 (0.13)	6.36 (0.06)
CIFAR-10 - $\epsilon = 4/255$							
1	(Grathwohl et al., 2020)	JEM-10	49.10 (0.09)	52.55 (0.06)	78.87 (0.16)	89.32 (0.06)	47.97 (0.05)
2	(Grathwohl et al., 2020)	JEM-1	46.08 (0.04)	49.71 (0.05)	78.93 (0.19)	90.17 (0.03)	45.49 (0.04)
3	(Grathwohl et al., 2020)	JEM-0	42.98 (0.10)	47.74 (0.08)	82.92 (0.10)	89.52 (0.13)	42.55 (0.07)

Table 9. **Comparison PGD vs APGD on the cross-entropy loss in the l_∞ -threat model.** We run PGD and PGD with Momentum (the same used for APGD) on the models tested in Sec. 6, with three different step sizes: $\epsilon/10$, $\epsilon/4$ and 2ϵ . Similarly to APGD, we use 100 iterations and 5 restarts for PGD, and report the resulting robust accuracy on the whole test set (1000 images for ImageNet). For each model we boldface the best attack and underline the best version of PGD (i.e. we exclude APGD).

#	paper	clean	PGD			PGD with Momentum			APGD
			$\epsilon/10$	$\epsilon/4$	2ϵ	$\epsilon/10$	$\epsilon/4$	2ϵ	
CIFAR-10 - $\epsilon = 8/255$									
1	(Carmon et al., 2019)	89.69	62.01	61.86	63.15	61.98	61.79	61.84	61.47
2	(Alayrac et al., 2019)	86.46	60.78	60.72	61.87	60.70	<u>60.58</u>	60.73	59.86
3	(Hendrycks et al., 2019)	87.11	57.24	<u>57.13</u>	57.88	57.28	57.14	<u>57.13</u>	57.00
4	(Rice et al., 2020)	85.34	56.87	56.82	57.55	56.88	<u>56.78</u>	56.81	56.76
5	(Qin et al., 2019)	86.28	55.50	55.48	56.42	55.50	<u>55.47</u>	55.62	55.45
6	(Engstrom et al., 2019)	87.03	52.06	51.90	53.30	52.08	51.87	<u>51.85</u>	51.52
7	(Kumari et al., 2019)	87.80	51.85	51.75	52.95	51.87	<u>51.65</u>	51.68	51.56
8	(Mao et al., 2019)	86.21	49.71	49.59	50.69	49.72	<u>49.56</u>	49.61	49.39
9	(Zhang et al., 2019a)	87.20	46.15	46.08	47.18	46.13	46.08	<u>46.04</u>	45.91
10	(Madry et al., 2018)	87.14	44.93	44.83	46.46	44.94	<u>44.81</u>	44.99	44.56
11	(Pang et al., 2020)	80.89	56.02	56.04	56.22	56.02	<u>56.01</u>	56.04	55.91
12	(Wong et al., 2020)	83.34	46.10	45.93	47.45	46.06	<u>45.90</u>	46.03	45.60
13	(Shafahi et al., 2019)	86.11	44.45	45.60	48.83	<u>43.63</u>	<u>44.50</u>	47.07	43.30
14	(Ding et al., 2020)	84.36	50.57	50.36	50.16	50.58	50.41	<u>49.66</u>	49.36
15	(Moosavi-Dezfooli et al., 2019)	83.11	41.76	41.71	42.33	41.70	41.68	41.58	41.59
16	(Zhang & Wang, 2019)	89.98	67.38	66.79	57.31	67.40	66.50	61.15	62.03
17	(Zhang & Xu, 2020)	90.25	71.69	71.42	62.52	71.77	71.48	68.88	69.36
18	(Jang et al., 2019)	78.91	38.00	37.88	<u>39.72</u>	37.90	<u>37.73</u>	37.90	37.40
19	(Kim & Wang, 2020)	91.51	59.16	58.24	51.29	59.12	<u>58.23</u>	54.17	54.80
20	(Moosavi-Dezfooli et al., 2019)	80.41	36.86	36.74	37.92	36.81	36.74	<u>36.61</u>	36.53
21	(Wang & Zhang, 2019)	92.80	61.92	60.61	54.96	62.05	60.66	56.89	57.19
22	(Wang & Zhang, 2019)	92.82	69.63	69.10	61.03	69.60	69.06	67.71	67.77
23	(Mustafa et al., 2019)	89.16	17.04	13.79	4.39	17.06	13.33	5.57	4.48
24	(Chan et al., 2020)	93.79	1.89	1.89	9.67	1.91	1.90	5.93	1.90
25	(Pang et al., 2020)	93.52	85.72	85.73	<u>85.67</u>	85.72	85.72	86.23	85.58
CIFAR-10 - $\epsilon = 0.031$									
1	(Zhang et al., 2019b)	84.92	55.28	55.24	56.24	55.25	<u>55.21</u>	55.30	55.08
2	(Atzmon et al., 2019)	81.30	78.94	78.94	78.94	78.94	78.94	78.94	78.94
3	(Xiao et al., 2020)	79.28	36.32	36.03	<u>34.19</u>	36.27	36.04	35.97	32.38
CIFAR-10 - $\epsilon = 4/255$									
1	(Song et al., 2019)	84.81	57.42	57.40	57.60	57.42	57.41	57.43	57.43
CIFAR-10 - $\epsilon = 0.02$									
1	(Pang et al., 2019)	91.22	7.27	5.96	17.23	7.00	<u>5.44</u>	9.52	3.53
2	(Pang et al., 2019)	93.44	0.36	0.24	3.37	0.41	<u>0.17</u>	1.22	0.04
CIFAR-100 - $\epsilon = 8/255$									
1	(Hendrycks et al., 2019)	59.23	33.01	32.91	33.17	33.02	32.83	32.76	32.83
2	(Rice et al., 2020)	53.83	20.61	20.57	21.08	20.57	20.52	<u>20.47</u>	20.32
MNIST - $\epsilon = 0.3$									
1	(Zhang et al., 2020)	98.38	95.25	95.23	95.21	95.06	<u>94.98</u>	94.99	94.58
2	(Gowal et al., 2019)	98.34	94.67	94.58	94.39	94.38	94.23	<u>94.19</u>	93.81
3	(Zhang et al., 2019b)	99.48	94.07	94.10	95.27	93.97	<u>93.80</u>	94.32	93.14
4	(Ding et al., 2020)	98.95	94.44	94.42	95.96	94.04	<u>93.53</u>	95.43	93.51
5	(Atzmon et al., 2019)	99.35	98.79	98.83	98.94	98.79	98.83	98.93	98.79
6	(Madry et al., 2018)	98.53	<u>90.75</u>	90.88	91.97	<u>90.37</u>	<u>90.29</u>	91.03	89.40
7	(Jang et al., 2019)	98.47	93.05	93.66	95.34	92.73	<u>92.47</u>	94.47	92.45
8	(Wong et al., 2020)	98.50	87.16	87.40	90.19	86.51	<u>86.30</u>	87.23	84.74
9	(Taghanaki et al., 2019)	98.86	20.61	20.27	20.23	20.56	<u>20.27</u>	20.43	23.83
ImageNet - $\epsilon = 4/255$									
1	(Engstrom et al., 2019)	63.4	31.5	<u>31.4</u>	34.0	31.5	<u>31.4</u>	31.9	30.9

Table 10. Comparison PGD vs APGD on the CW loss in the l_∞ -threat model. We run PGD and PGD with Momentum (the same used for APGD) on the models tested in Sec. 6, with three different step sizes: $\epsilon/10$, $\epsilon/4$ and 2ϵ . Similarly to APGD, we use 100 iterations and 5 restarts for PGD, and report the resulting robust accuracy on the whole test set (1000 images for ImageNet). For each model we boldface the best attack and underline the best version of PGD (i.e. we exclude APGD).

#	paper	clean	PGD			PGD with Momentum			APGD
			$\epsilon/10$	$\epsilon/4$	2ϵ	$\epsilon/10$	$\epsilon/4$	2ϵ	
CIFAR-10 - $\epsilon = 8/255$									
1	(Carmon et al., 2019)	89.69	60.73	60.72	62.20	60.73	60.69	61.02	60.48
2	(Alayrac et al., 2019)	86.46	61.78	61.85	63.23	61.69	61.61	62.10	61.13
3	(Hendrycks et al., 2019)	87.11	56.44	56.42	57.15	56.45	56.41	56.39	56.20
4	(Rice et al., 2020)	85.34	55.24	55.21	56.04	55.23	55.19	55.26	55.07
5	(Qin et al., 2019)	86.28	55.10	55.10	55.95	55.10	55.11	55.22	54.99
6	(Engstrom et al., 2019)	87.03	52.24	52.17	53.61	52.26	52.16	52.36	51.94
7	(Kumari et al., 2019)	87.80	51.13	51.07	52.67	51.14	51.03	51.22	50.88
8	(Mao et al., 2019)	86.21	49.88	49.84	51.18	49.88	49.84	50.02	49.71
9	(Zhang et al., 2019a)	87.20	47.13	47.09	48.27	47.11	47.06	47.08	46.89
10	(Madry et al., 2018)	87.14	45.98	45.82	47.47	45.95	45.77	46.11	45.67
11	(Pang et al., 2020)	80.89	44.40	44.41	45.83	44.41	44.36	44.66	44.11
12	(Wong et al., 2020)	83.34	46.04	45.94	47.72	46.03	45.93	46.26	45.66
13	(Shafahi et al., 2019)	86.11	44.89	45.93	49.27	44.16	45.10	47.43	43.90
14	(Ding et al., 2020)	84.36	51.60	51.44	51.09	51.59	51.29	50.63	50.25
15	(Moosavi-Dezfooli et al., 2019)	83.11	40.21	40.18	40.93	40.20	40.16	40.22	40.10
16	(Zhang & Wang, 2019)	89.98	55.98	55.30	48.59	55.99	55.27	50.91	51.65
17	(Zhang & Xu, 2020)	90.25	67.00	66.61	57.87	67.02	66.67	64.19	64.46
18	(Jang et al., 2019)	78.91	36.85	36.81	39.32	36.85	36.83	37.30	36.52
19	(Kim & Wang, 2020)	91.51	56.85	55.90	50.26	56.94	55.92	52.54	53.04
20	(Moosavi-Dezfooli et al., 2019)	80.41	35.38	35.37	36.86	35.38	35.35	35.48	35.29
21	(Wang & Zhang, 2019)	92.80	59.93	58.42	52.56	60.01	58.46	54.51	54.77
22	(Wang & Zhang, 2019)	92.82	66.49	65.86	56.71	66.60	65.76	63.53	63.80
23	(Mustafa et al., 2019)	89.16	18.27	14.75	4.42	18.44	14.66	5.46	4.73
24	(Chan et al., 2020)	93.79	1.32	1.28	5.51	1.30	1.29	1.29	1.23
25	(Pang et al., 2020)	93.52	1.88	0.54	1.42	1.59	0.53	0.47	0.11
CIFAR-10 - $\epsilon = 0.031$									
1	(Zhang et al., 2019b)	84.92	54.05	54.06	55.16	54.08	54.03	54.22	53.90
2	(Atzmon et al., 2019)	81.30	40.37	40.28	42.30	40.36	40.21	40.56	40.05
3	(Xiao et al., 2020)	79.28	35.77	35.53	34.04	36.14	35.31	35.76	32.09
CIFAR-10 - $\epsilon = 4/255$									
1	(Song et al., 2019)	84.81	57.54	57.53	57.78	57.52	57.53	57.51	57.49
CIFAR-10 - $\epsilon = 0.02$									
1	(Pang et al., 2019)	91.22	7.12	6.09	16.15	6.87	5.57	9.49	3.82
2	(Pang et al., 2019)	93.44	0.47	0.26	3.13	0.42	0.13	1.22	0.12
CIFAR-100 - $\epsilon = 8/255$									
1	(Hendrycks et al., 2019)	59.23	30.64	30.61	31.19	30.63	30.61	30.56	30.45
2	(Rice et al., 2020)	53.83	20.22	20.21	21.06	20.22	20.22	20.35	19.98
MNIST - $\epsilon = 0.3$									
1	(Zhang et al., 2020)	98.38	95.22	95.21	95.17	95.10	94.97	95.04	94.61
2	(Gowal et al., 2019)	98.34	94.63	94.48	94.39	94.49	94.12	94.23	93.58
3	(Zhang et al., 2019b)	99.48	94.40	94.35	95.23	94.18	94.04	94.33	93.35
4	(Ding et al., 2020)	98.95	94.49	94.52	95.85	93.98	93.81	95.38	93.53
5	(Atzmon et al., 2019)	99.35	94.43	94.69	95.98	94.03	93.74	95.21	92.98
6	(Madry et al., 2018)	98.53	90.82	90.96	91.94	90.39	90.30	91.07	89.39
7	(Jang et al., 2019)	98.47	93.28	93.64	95.26	92.68	92.71	94.49	92.40
8	(Wong et al., 2020)	98.50	87.32	87.55	90.20	86.74	86.62	87.25	84.97
9	(Taghanaki et al., 2019)	98.86	0.00	0.00	0.00	0.00	0.00	0.00	0.00
ImageNet - $\epsilon = 4/255$									
1	(Engstrom et al., 2019)	63.4	31.7	31.6	34.5	31.7	31.7	32.5	31.5

Table 11. Comparison PGD vs APGD on the DLR loss in the l_∞ -threat model. We run PGD and PGD with Momentum (the same used for APGD) on the models tested in Sec. 6, with three different step sizes: $\epsilon/10$, $\epsilon/4$ and 2ϵ . Similarly to APGD, we use 100 iterations and 5 restarts for PGD, and report the resulting robust accuracy on the whole test set (1000 images for ImageNet). For each model we boldface the best attack and underline the best version of PGD (i.e. we exclude APGD).

#	paper	clean	PGD			PGD with Momentum			APGD
			$\epsilon/10$	$\epsilon/4$	2ϵ	$\epsilon/10$	$\epsilon/4$	2ϵ	
CIFAR-10 - $\epsilon = 8/255$									
1	(Carmon et al., 2019)	89.69	60.98	60.88	62.41	60.93	<u>60.82</u>	61.25	60.64
2	(Alayrac et al., 2019)	86.46	63.27	63.15	64.71	63.16	<u>63.00</u>	63.55	62.03
3	(Hendrycks et al., 2019)	87.11	57.27	<u>57.18</u>	57.98	57.27	<u>57.18</u>	57.22	56.96
4	(Rice et al., 2020)	85.34	55.89	<u>55.86</u>	56.75	55.90	<u>55.85</u>	55.93	55.72
5	(Qin et al., 2019)	86.28	55.73	55.70	56.47	55.73	<u>55.68</u>	55.70	55.46
6	(Engstrom et al., 2019)	87.03	53.13	53.04	54.41	53.09	<u>52.95</u>	53.29	52.62
7	(Kumari et al., 2019)	87.80	51.97	51.92	53.58	51.98	<u>51.89</u>	52.11	51.68
8	(Mao et al., 2019)	86.21	50.79	50.57	52.03	50.74	<u>50.52</u>	50.79	50.33
9	(Zhang et al., 2019a)	87.20	47.70	47.58	48.84	47.71	47.57	<u>47.53</u>	47.33
10	(Madry et al., 2018)	87.14	46.49	<u>46.30</u>	48.13	46.51	<u>46.30</u>	46.58	46.03
11	(Pang et al., 2020)	80.89	44.97	44.91	46.36	44.96	<u>44.87</u>	45.21	44.56
12	(Wong et al., 2020)	83.34	47.20	47.05	48.62	47.18	<u>47.01</u>	47.22	46.64
13	(Shafahi et al., 2019)	86.11	45.46	46.43	50.03	44.74	<u>45.51</u>	47.89	44.56
14	(Ding et al., 2020)	84.36	51.67	51.44	51.46	51.68	51.39	<u>50.93</u>	50.32
15	(Moosavi-Dezfooli et al., 2019)	83.11	40.36	40.33	41.29	40.37	<u>40.32</u>	40.43	40.29
16	(Zhang & Wang, 2019)	89.98	55.69	54.73	46.57	55.68	54.68	48.53	48.96
17	(Zhang & Xu, 2020)	90.25	59.16	56.77	46.72	59.13	56.61	49.07	49.43
18	(Jang et al., 2019)	78.91	37.41	37.30	39.77	37.38	<u>37.27</u>	37.69	37.01
19	(Kim & Wang, 2020)	91.51	52.42	51.41	48.09	52.43	51.30	48.51	48.41
20	(Moosavi-Dezfooli et al., 2019)	80.41	35.57	35.53	37.25	35.57	<u>35.52</u>	35.68	35.47
21	(Wang & Zhang, 2019)	92.80	61.40	53.79	38.23	61.31	53.27	39.16	40.69
22	(Wang & Zhang, 2019)	92.82	54.84	49.01	35.15	54.83	48.62	36.05	36.72
23	(Mustafa et al., 2019)	89.16	22.16	17.10	<u>4.90</u>	22.18	17.42	5.57	4.54
24	(Chan et al., 2020)	93.79	9.52	7.11	10.41	7.56	5.38	5.30	1.20
25	(Pang et al., 2020)	93.52	1.94	0.47	1.75	2.06	0.58	0.75	0.49
CIFAR-10 - $\epsilon = 0.031$									
1	(Zhang et al., 2019b)	84.92	54.25	54.20	55.28	54.23	<u>54.16</u>	54.34	54.04
2	(Atzmon et al., 2019)	81.30	57.45	51.99	44.62	56.19	50.03	44.45	44.50
3	(Xiao et al., 2020)	79.28	37.74	36.33	<u>33.46</u>	37.49	36.92	35.80	31.27
CIFAR-10 - $\epsilon = 4/255$									
1	(Song et al., 2019)	84.81	57.85	57.83	58.07	57.84	57.83	<u>57.81</u>	57.79
CIFAR-10 - $\epsilon = 0.02$									
1	(Pang et al., 2019)	91.22	15.83	13.33	18.98	15.23	<u>12.38</u>	16.31	8.61
2	(Pang et al., 2019)	93.44	1.53	0.98	3.58	1.25	<u>0.68</u>	2.59	0.26
CIFAR-100 - $\epsilon = 8/255$									
1	(Hendrycks et al., 2019)	59.23	32.05	31.91	32.14	32.03	31.92	31.70	31.68
2	(Rice et al., 2020)	53.83	20.41	20.34	21.41	20.40	<u>20.33</u>	20.47	20.20
MNIST - $\epsilon = 0.3$									
1	(Zhang et al., 2020)	98.38	95.65	95.52	95.26	95.41	95.17	<u>95.14</u>	94.82
2	(Gowal et al., 2019)	98.34	94.93	94.73	94.28	94.72	94.45	<u>94.22</u>	93.87
3	(Zhang et al., 2019b)	99.48	94.84	94.78	95.53	94.63	<u>94.50</u>	94.79	93.89
4	(Ding et al., 2020)	98.95	94.69	94.78	95.80	94.16	<u>94.00</u>	95.40	93.86
5	(Atzmon et al., 2019)	99.35	94.70	94.99	96.32	94.10	93.95	95.53	94.54
6	(Madry et al., 2018)	98.53	91.08	91.21	92.15	90.65	<u>90.51</u>	91.51	89.74
7	(Jang et al., 2019)	98.47	93.19	93.79	94.74	<u>92.72</u>	92.99	94.49	92.15
8	(Wong et al., 2020)	98.50	87.58	87.65	90.40	86.99	<u>86.87</u>	87.74	85.39
9	(Taghanaki et al., 2019)	98.86	0.00	0.00	0.00	0.00	0.00	0.00	0.00
ImageNet - $\epsilon = 4/255$									
1	(Engstrom et al., 2019)	63.4	<u>32.1</u>	<u>32.1</u>	35.0	<u>32.1</u>	<u>32.1</u>	32.8	32.0

Table 12. Comparison PGD vs APGD on the cross-entropy loss in the l_2 -threat model. We run PGD and PGD with Momentum (the same used for APGD) on the models tested in Sec. 6, with three different step sizes: $\epsilon/10$, $\epsilon/4$ and 2ϵ . Similarly to APGD, we use 100 iterations and 5 restarts for PGD, and report the resulting robust accuracy on the whole test set (1000 images for ImageNet). For each model we boldface the best attack and underline the best version of PGD (i.e. we exclude APGD).

#	paper	clean	PGD			PGD with Momentum			APGD
			$\epsilon/10$	$\epsilon/4$	2ϵ	$\epsilon/10$	$\epsilon/4$	2ϵ	
CIFAR-10 - $\epsilon = 0.5$									
1	(Augustin et al., 2020)	91.08	74.81	74.74	74.60	74.81	74.75	74.63	74.65
2	(Engstrom et al., 2019)	90.83	69.68	69.65	69.58	69.69	69.63	69.62	69.60
3	(Rice et al., 2020)	88.67	68.64	68.60	<u>68.55</u>	68.64	68.58	68.56	68.53
4	(Rony et al., 2019)	89.05	66.60	<u>66.58</u>	66.63	66.59	<u>66.58</u>	<u>66.58</u>	66.57
5	(Ding et al., 2020)	88.02	66.22	<u>66.21</u>	66.29	66.22	<u>66.21</u>	<u>66.21</u>	66.19
ImageNet - $\epsilon = 3$									
1	(Engstrom et al., 2019)	55.3	31.9	31.6	31.6	31.9	31.5	31.6	31.5

Table 13. Comparison PGD vs APGD on the CW loss in the l_2 -threat model. We run PGD and PGD with Momentum (the same used for APGD) on the models tested in Sec. 6, with three different step sizes: $\epsilon/10$, $\epsilon/4$ and 2ϵ . Similarly to APGD, we use 100 iterations and 5 restarts for PGD, and report the resulting robust accuracy on the whole test set (1000 images for ImageNet). For each model we boldface the best attack and underline the best version of PGD (i.e. we exclude APGD).

#	paper	clean	PGD			PGD with Momentum			APGD
			$\epsilon/10$	$\epsilon/4$	2ϵ	$\epsilon/10$	$\epsilon/4$	2ϵ	
CIFAR-10 - $\epsilon = 0.5$									
1	(Augustin et al., 2020)	91.08	74.57	74.54	74.55	74.56	74.55	74.53	74.49
2	(Engstrom et al., 2019)	90.83	70.08	70.07	70.06	70.08	70.07	70.06	70.06
3	(Rice et al., 2020)	88.67	68.73	68.71	68.72	68.73	68.72	68.71	68.71
4	(Rony et al., 2019)	89.05	67.00	67.00	67.03	67.00	67.00	66.99	66.99
5	(Ding et al., 2020)	88.02	66.55	66.54	66.64	66.55	66.54	66.53	66.54
ImageNet - $\epsilon = 3$									
1	(Engstrom et al., 2019)	55.3	30.7	30.7	30.8	30.8	30.7	30.7	30.7

Table 14. Comparison PGD vs APGD on the DLR loss in the l_2 -threat model. We run PGD and PGD with Momentum (the same used for APGD) on the models tested in Sec. 6, with three different step sizes: $\epsilon/10$, $\epsilon/4$ and 2ϵ . Similarly to APGD, we use 100 iterations and 5 restarts for PGD, and report the resulting robust accuracy on the whole test set (1000 images for ImageNet). For each model we boldface the best attack and underline the best version of PGD (i.e. we exclude APGD).

#	paper	clean	PGD			PGD with Momentum			APGD
			$\epsilon/10$	$\epsilon/4$	2ϵ	$\epsilon/10$	$\epsilon/4$	2ϵ	
CIFAR-10 - $\epsilon = 0.5$									
1	(Augustin et al., 2020)	91.08	75.06	75.00	<u>74.99</u>	75.06	75.00	<u>74.99</u>	74.94
2	(Engstrom et al., 2019)	90.83	70.20	70.19	70.17	70.20	70.19	<u>70.17</u>	70.20
3	(Rice et al., 2020)	88.67	69.02	69.01	68.96	69.02	69.02	<u>68.94</u>	68.95
4	(Rony et al., 2019)	89.05	67.03	67.02	67.06	67.03	67.02	<u>67.02</u>	67.02
5	(Ding et al., 2020)	88.02	66.61	<u>66.56</u>	66.64	66.61	<u>66.55</u>	<u>66.54</u>	66.53
ImageNet - $\epsilon = 3$									
1	(Engstrom et al., 2019)	55.3	30.9	<u>30.9</u>	<u>30.9</u>	<u>30.9</u>	31.1	<u>30.9</u>	30.9