Mind the Box: l_1-APGD for Sparse Adversarial Attacks on Image Classifiers

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

Abstract

We show that when taking into account also the image domain $[0, 1]^d$, established l_1-projected gradient descent (PGD) attacks are suboptimal as they do not consider that the effective threat model is the intersection of the l_1-ball and $[0, 1]^d$. We study the expected sparsity of the steepest descent step for this effective threat model and show that the exact projection onto this set is computationally feasible and yields better performance. Moreover, we propose an adaptive form of PGD which is highly effective even with a small budget of iterations. Our resulting l_1-APGD is a strong white-box attack showing that prior works overestimated their l_1-robustness. Using l_1-APGD for adversarial training we get a robust classifier with SOTA l_1-robustness. Finally, we combine l_1-APGD and an adaptation of the Square Attack to l_1 into l_1-AutoAttack, an ensemble of attacks which reliably assesses adversarial robustness for the threat model of l_1-ball intersected with $[0, 1]^d$.

1. Introduction

The application of machine learning in safety-critical systems requires reliable decisions. Small adversarial perturbations (Szegedy et al., 2014; Kurakin et al., 2017), changing the decision of a classifier, without changing the semantic content of the image are a major problem. While adversarial training (Madry et al., 2018) and recent variations and improvements (Carmon et al., 2019; Gowal et al., 2020; Wu et al., 2021) are a significant progress, most proposed defenses not involving some form of adversarial training turn out to be non-robust (Carlini & Wagner, 2017; Athalye et al., 2018). While the community so far has focused mainly on l_∞- and l_2-perturbations, l_1-perturbation sets are complementary as they lead to very sparse changes which leave effectively most of the image unmodified and thus should also not lead to a change in the decision. While there exist a set of l_1-based attacks (Chen et al., 2018; Modas et al., 2019; Brendel et al., 2019; Croce & Hein, 2020a; Rony et al., 2020), in contrast to the l_∞- and l_2-case the classical white-box projected gradient descent (PGD) attack of (Madry et al., 2018) has not an established standard form (Tramèr & Boneh, 2019; Maini et al., 2020). Moreover, training l_1-robust models with adversarial training has been reported to be difficult (Maini et al., 2020; Liu et al., 2020).

In this paper we identify reasons why the current versions of l_1-PGD attacks are weaker than SOTA l_1-attacks (Chen et al., 2018; Croce & Hein, 2020a; Rony et al., 2020). A key issue is that in image classification we have the additional constraint that the input has to lie in the box $[0, 1]^d$ and thus the effective threat model is the intersection of the l_1-ball and $[0, 1]^d$. However, current l_1-PGD attacks only approximate the correct projection onto this set (Tramèr & Boneh, 2019) and argue for a steepest descent direction without taking into account the box constraints. We first show that the correct projection onto the intersection can be computed in essentially the same time as the projection onto the l_1-ball, and then we discuss theoretically and empirically that using the approximate projection leads to a worse attack as it cannot access certain parts of the threat model. Moreover, we derive the correct steepest descent step for the intersection of l_1-ball and $[0, 1]^d$ which motivates an adaptive sparsity of the chosen descent direction. Then, inspired by the recent work on Auto-PGD (APGD) (Croce & Hein, 2020b) for l_2 and l_∞, we design a novel fully adaptive parameter-free PGD scheme so that the user does not need to do step size selection for each defense separately which is known to be error prone. Interestingly, using our l_1-APGD we are able to train the model with the highest l_1-robust accuracy for $\epsilon = 12$ while standard PGD fails due to catastrophic overfitting (Wong et al., 2020) and/or overfitting to the sparsity of the standard PGD attack. Finally, following (Croce & Hein, 2020b) we assemble l_1-APGD for two different losses, the targeted l_1-FAB attack (Croce & Hein, 2020a), and an l_1-adaptation of the SOTA black-box Square Attack (Andriushchenko et al., 2020) into a novel parameter-free l_1-AutoAttack which leads to a reliable and effective assessment of l_1-robustness similar to AutoAttack (Croce & Hein, 2020b) for the l_2- and l_∞-case. All proofs can be found in App. A.

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

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

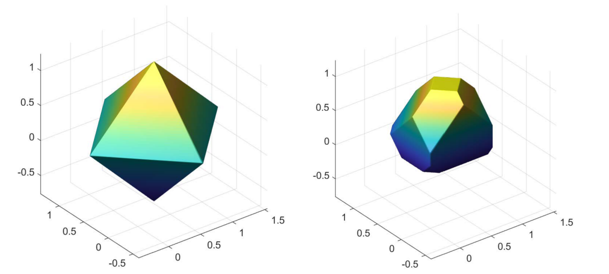

Figure 1. Left: the l_1-ball $B_1(x, 1)$ centered at the (randomly chosen) target point $x \in [0, 1]^3$, right: the intersection $S = B_1(x, 1) \cap [0, 1]^3$ of $B_1(x, 1)$ with the box $[0, 1]^3$.

2. Mind the box $[0, 1]^d$ in l_1-PGD

Projected Gradient Descent is a simple first-order method which consists in a descent step followed by a projection onto the feasible set S, that is, given the current iterate $x^{(i)}$, the next iterate $x^{(i+1)}$ is computed as

$$u^{(i+1)} = x^{(i)} + \eta^{(i)} \cdot s(\nabla L(x^{(i)})), \tag{1}$$
$$x^{(i+1)} = P_S(u^{(i+1)}), \tag{2}$$

where d is the input dimension, $\eta^{(i)} > 0$ the step size at iteration i, $s : \mathbb{R}^d \to \mathbb{R}^d$ determines the descent direction as a function of the gradient of the loss L at $x^{(i)}$ and $P_S : \mathbb{R}^d \to S$ is the projection on S. With an l_1-perturbation model of radius ϵ, we denote by $B_1(x, \epsilon) := \{z \in \mathbb{R}^d \mid \|z - x\|_1 \leq \epsilon\}$ the l_1-ball around a target point $x \in [0, 1]^d$ and define $S = [0, 1]^d \cap B_1(x, \epsilon)$. The main difference to prior work is that we take explicitly into account the image constraint $[0, 1]^d$. Note that the geometry of the effective threat model is actually quite different from $B_1(x, \epsilon)$ alone, see Figure 1 for an illustration. In the following we analyse the projection and the descent step in this effective threat model S. As ϵ is significantly higher for l_1 (we use $\epsilon = 12$ similar to (Maini et al., 2020)) as for l_2 (standard 0.5) and l_∞ (standard $\frac{8}{255}$) the difference of the intersection with $[0, 1]^d$ to the l_p-ball alone is most prominent for the l_1-case.

2.1. Projection onto S

With $B_1(x, \epsilon)$ as defined above and denoting $H = [0, 1]^d$ the image box, we consider the two projection problems:

$$P_S(u) = \operatorname*{arg\,max}_{z \in \mathbb{R}^d} \|u - z\|_2^2 \qquad \text{s.th.} \quad \|z - x\|_1 \leq \epsilon, \quad z \in [0, 1]^d. \tag{3}$$

and

$$P_{B_1(x,\epsilon)}(u) = \operatorname*{arg\,max}_{z \in \mathbb{R}^d} \|u - z\|_2^2 \quad \text{s.th.} \quad \|z - x\|_1 \leq \epsilon. \tag{4}$$

It is well known that the l_1-projection problem in (4) can be solved in $O(d \log d)$ (Duchi et al., 2008; Condat, 2016). We show now that also the exact projection onto S can be computed with the same complexity (after this paper has been accepted we got aware of (Wang et al., 2019) who derived also the form of the solution of (3) but provided no complexity analysis or an algorithm to compute it).

Proposition 2.1 *The projection problem (3) onto $S = B_1(x, \epsilon) \cap H$ can be solved in $O(d \log d)$ with solution*

$$z_i^* = \begin{cases} 1 & \text{for } u_i \geq x_i \text{ and } 0 \leq \lambda_e^* \leq u_i - 1 \\ u_i - \lambda_e^* & \text{for } u_i \geq x_i \text{ and } u_i - 1 < \lambda_e^* \leq u_i - x_i \\ x_i & \text{for } \lambda_e^* > |u_i - x_i| \\ u_i + \lambda_e^* & \text{for } u_i \leq x_i \text{ and } -u_i < \lambda_e^* \leq x_i - u_i \\ 0 & \text{for } u_i \leq x_i \text{ and } 0 \leq \lambda_e^* \leq -u_i \end{cases},$$

where $\lambda_e^* \geq 0$. With $\gamma \in \mathbb{R}^d$ defined as

$$\gamma_i = \max\{-x_i \operatorname{sign}(u_i - x_i), (1 - x_i)\operatorname{sign}(u_i - x_i)\},$$

it holds $\lambda_e^* = 0$ if $\sum_{i=1}^d \max\{0, \min\{|u_i - x_i|, \gamma_i\} \leq \epsilon$ and otherwise λ_e^* is the solution of

$$\sum_{i=1}^d \max\left\{0, \min\{|u_i - x_i| - \lambda_e^*, \gamma_i\}\right\} = \epsilon.$$

The two prior versions of PGD (Tramèr & Boneh, 2019; Maini et al., 2020) for the l_1-threat model use the approximation $A : \mathbb{R}^d \to S$

$$A(u) = (P_H \circ P_{B_1(x,\epsilon)})(u),$$

instead of the exact projection $P_S(u)$ (see the appendix for a proof that $A(u) \in S$ for any $u \in \mathbb{R}^d$). However, it turns out that the approximation $A(u)$ "hides" parts of S due to the following property.

Lemma 2.1 *It holds for any $u \in \mathbb{R}^d$,*

$$\|P_S(u) - x\|_1 \geq \|A(u) - x\|_1 .$$

In particular, if $P_{B_1(x,\epsilon)}(u) \notin H$ and $\|u - x\|_1 > \epsilon$ and one of the following conditions holds

- $\|P_S(u) - x\|_1 = \epsilon$
- $\|P_S(u) - x\|_1 < \epsilon$ *and* $\exists u_i \in [0, 1]$ *with* $u_i \neq x_i$

then

$$\|P_S(u) - x\|_1 > \|A(u) - x\|_1 .$$

The previous lemma shows that the approximation $A(u)$ of $P_S(u)$ used by (Maini et al., 2020; Tramèr & Boneh, 2019)

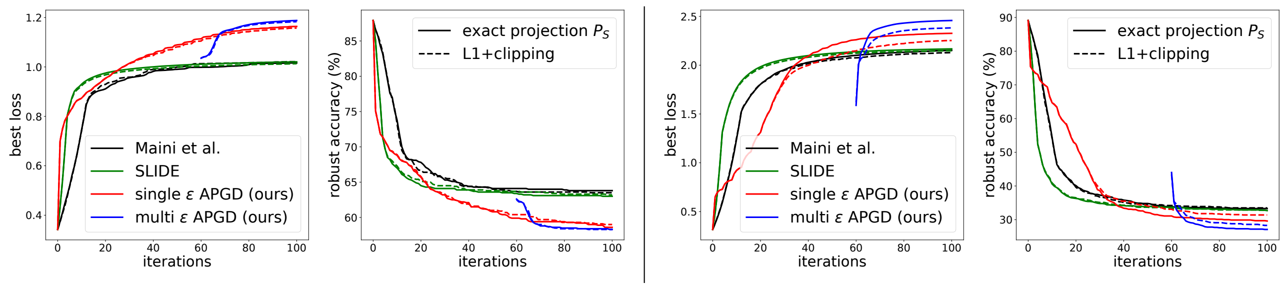

Figure 2. Plots of the best robust loss obtained so far (first/third) and robust accuracy (second/fourth) as a function of the iterations for the l_1-PGD of (Tramèr & Boneh, 2019) (SLIDE), the one of (Maini et al., 2020) and our single ϵ-APGD and multi ϵ-APGD for two models (left: our own l_1-robust model APGD-AT, right: the l_2-robust model of (Rice et al., 2020)). All of them are once run with the correct projection $P_S(u)$ (solid) and once with the approximation $A(u)$ (dashed). The exact projection improves in almost all cases for all attacks loss and robust accuracy. Moreover, our single- and multi-ϵ APGD improve significantly the robust loss as well as robust accuracy over SLIDE and the l_1-PGD of (Maini et al., 2020). Multi ϵ-APGD is only partially plotted as only the last 40% of iterations are feasible.

is definitely suboptimal under relatively weak conditions and has a smaller l_1-distance to the target point x:

$$\|P_S(u) - x\|_1 > \|A(u) - x\|_1 .$$

Effectively, a part of S is hidden from the attack when $A(u)$ instead of $P_S(u)$ is used (see Figure 4 in App. A for a practical example of this phenomenon). This in turn leads to suboptimal performance both in the maximization of the loss which is important for adversarial training but also in terms of getting low robust accuracy: the plots in Figure 2 show the performance of PGD-based attacks with $A(u)$ (dashed line) vs the same methods with the correct projection $P_S(u)$ (solid line). Our proposed l_1-APGD largely benefits from using $P_S(u)$ instead of $A(u)$, and this even slightly improves the existing l_1-versions of PGD, SLIDE (Tramèr & Boneh, 2019) and the one of (Maini et al., 2020). Thus we use in our scheme always the correct projection onto S (in the appendix more statistics on the difference of $A(u)$ and $P_S(u)$).

2.2. Descent direction

The next crucial step in the PGD scheme in (1) is the choice of the descent direction which we wrote as the mapping $s(\nabla f(x_i))$ of the gradient. For the l_∞- and l_2-threat models (Madry et al., 2018) the steepest descent direction (Boyd & Vandenberhge, 2004) is used in PGD, that is

$$\delta_p^* = \operatorname*{arg\,max}_{\delta \in \mathbb{R}^d} \langle w, \delta\rangle \quad \text{s.th.} \quad \|\delta\|_p \leq \epsilon, \tag{5}$$

with $w = \nabla f(x^{(i)}) \in \mathbb{R}^d$, which maximizes a linear function over the given l_p-ball. Thus one gets $\delta_\infty^* = \epsilon \operatorname{sign}(w)$ and $\delta_2^* = \epsilon w / \|w\|_2$ for $p = \infty$ and $p = 2$ respectively, which define the function s in (1). For $p = 1$, defining $j = \operatorname*{arg\,max}_i |w_i|$ the dimension corresponding to the component of w with largest absolute value and $\mathcal{B} = \{e_i\}_i$ the standard basis of $\mathbb{R}^d$, we have $\delta_1^* = \epsilon \operatorname{sign}(w_j) e_j$. Obviously, for a small number of iterations this descent direction is not working well and thus in SLIDE (Tramèr & Boneh, 2019) suggest to use the top-k components of the gradient (ordered according to their magnitude) and use the sign of these components.

In the following we show that when one takes into account the box-constraints imposed by the image domain the steepest descent direction becomes automatically less sparse and justifies at least partially what has been done in SLIDE (Tramèr & Boneh, 2019) and (Maini et al., 2020) out of efficiency reasons. More precisely, the following optimization problem defines the steepest descent direction:

$$\delta^* = \operatorname*{arg\,max}_{\delta \in \mathbb{R}^d} \langle w, \delta\rangle \tag{6}$$
$$\text{s.th.} \quad \|\delta\|_1 \leq \epsilon, \quad x + \delta \in [0,1]^d.$$

Proposition 2.2 *Let* $z_i = \max\{(1 - x_i)\operatorname{sign}(w_i), -x_i \operatorname{sign}(w_i)\}$, π *the ordering such that* $|w_{\pi_i}| \geq |w_{\pi_j}|$ *for* $i > j$ *and* k *the smallest integer for which* $\sum_{i=1}^k z_{\pi_i} \geq \epsilon$*, then the solution of* (6) *is given by*

$$\delta_{\pi_i}^* = \begin{cases} z_{\pi_i} \cdot \operatorname{sign}(w_{\pi_i}) & \textit{for } i < k, \\ (\epsilon - \sum_{i=1}^{k-1} z_{\pi_i}) \cdot \operatorname{sign}(w_{\pi_k}) & \textit{for } i = k, \\ 0 & \textit{for } i > k \end{cases} . \tag{7}$$

Proposition 2.2 shows that adding the box-constraints leads to a steepest descent direction δ^* of sparsity level k which depends on the gradient direction w and the target point x. Figure 5 in App. A provides an empirical evaluation of the distribution of the sparsity level $\|\delta^*\|_0$. The following proposition computes the expected sparsity of the steepest descent step δ^* for $\epsilon \leq \frac{d-1}{2}$, together with a simple lower bound.

Proposition 2.3 *Let $w \in \mathbb{R}^d$ with $w_i \neq 0$ for all $i = 1, \dots, d$ and $x \in \mathcal{U}([0,1]^d)$. Then it holds for any $\frac{d-1}{2} \geq \epsilon > 0$,*

$$\mathbb{E}\big[\, \|\delta^*\|_0 \,\big] = \lfloor \epsilon + 1 \rfloor + \sum_{m=\lfloor \epsilon \rfloor + 2}^{d} \sum_{k=0}^{\lfloor \epsilon \rfloor} (-1)^k \frac{(\epsilon - k)^{m-1}}{k!\,(m-1-k)!} \geq \frac{\lfloor 3\epsilon \rfloor + 1}{2}.$$

While the exact expression is hard to access, the derived lower bound $\frac{\lfloor 3\epsilon \rfloor - 1}{2}$ shows that the sparsity is non-trivially bounded away from 1. For a reasonable range of ϵ the expectation is numerically larger than 2ϵ. In this way we provide a justification for the heuristic non-sparse update steps used in (Tramèr & Boneh, 2019; Maini et al., 2020).

Finally, in our PGD scheme given $g = \nabla L(x^{(i)})$, $t \in \mathbb{N}$ and $T(t)$ the set of indices of the t largest components of $|g|$, we define the function s used in (1) via

$$h(t)_i = \begin{cases} \mathrm{sign}(g_i) & \text{if } i \in T(t) \\ 0 & \text{else} \end{cases}, \quad s(g,t) = h(t) / \|h(t)\|_1 \tag{8}$$

defines the function s used in (1). The form of the update is the same as in SLIDE (Tramèr & Boneh, 2019) who use a fixed k. However, as derived above, the sparsity level k of the steepest descent direction depends on ∇f and x and thus we choose k our scheme in a dynamic fashion depending on the current iterate, as described in the next section.

3. l_1-APGD minds $[0,1]^d$

The goal of our l_1-APGD is similar to that of APGD for l_2/l_∞ in (Croce & Hein, 2020b). It should be parameter-free for the user and adapt the trade-off between exploration and local fine-tuning to the given budget of iterations.

Proposition 2.2 suggests the form of the steepest descent direction for the l_1-ball $\cap\, [0,1]^d$ threat model, which has an expected sparsity on the order of 2ϵ but the optimal sparsity depends on the target point and the gradient of the loss. Thus fixing the sparsity of the update independent of the target point as in SLIDE (Tramèr & Boneh, 2019) is suboptimal. In practice we have $\epsilon \ll d$ (e.g. for CIFAR-10 $d = 3072$ and commonly $\epsilon = 12$) and thus 2ϵ sparse updates would lead to slow progress which is in strong contrast to the tight iteration budget used in adversarial attacks. Thus we need a scheme where the sparsity is adaptive to the chosen budget of iterations and depends on the current iterate. This motivates two key choices in our scheme: 1) we start with updates with low sparsity, 2) the sparsity of the updates is then progressively reduced and adapted to the sparsity of the difference of our currently best iterate (highest loss) to the target point. Thus, initially many coordinates are updated fostering fast progress and exploration of the feasible set, while later on we have a more local exploitation with significantly sparser updates. We observe that, although we do not enforce this actively, the average (over points) sparsity of the updates selected by our adaptive scheme towards the final iterations is indeed on the order of 2ϵ, i.e. around to what theoretically expected, although the exact value varies across models. In the following we describe the details of our l_1-APGD, see Algorithm 1, and a multi-ϵ variant which increases the effectiveness, and finally discuss its use for adversarial training (Madry et al., 2018).

Algorithm 1 Single-ϵ APGD

Input: loss L, initial point x_{init}, feasible set S, N_{iter}, $\eta^{(0)}$, $k^{(0)}$, checkpoints M, input dimension d
Output: approximate maximizer of the loss x_{best}
$x^{(0)} \leftarrow x_{\text{init}}$, $x_{\text{best}} \leftarrow x_{\text{init}}$, $L_{\text{best}} \leftarrow L(x_{\text{init}})$
for $i = 0$ **to** N_{iter} - 1 **do**
    // adjust sparsity and step size
    if $i + 1 \in M$ **then**
        $k^{(i+1)} \leftarrow$ sparsity as in Eq. (9)
        $\eta^{(i+1)} \leftarrow$ step size as in Eq. (10)
        if $\eta^{(i+1)} = \eta^{(0)}$ **then**
            $x^{(i)} \leftarrow x_{\text{best}}$
        end if
    end if
    // update step
    $u^{(i+1)} = x^{(i)} + \eta^{(i)} \cdot s(\nabla L(x^{(i)}), k^{(i+1)} \cdot d)$
    $x^{(i+1)} = P_S(u^{(i+1)})$
    // update best point found
    if $L(x^{(i+1)}) > L_{\text{best}}$ **then**
        $x_{\text{best}} \leftarrow x^{(i+1)}$, $L_{\text{best}} \leftarrow L(x^{(i+1)})$
    end if
end for

3.1. Single-ϵ l_1-APGD

Our scheme should automatically adapt to the total budget of iterations. Since the two main quantities which control the optimization in the intersection of l_1-ball and $[0,1]^d$ are the sparsity of the updates and the step size, we propose to adaptively select them at each iteration. In particular, we adjust both every $m = \lceil 0.04 \cdot N_{\text{iter}} \rceil$ steps, with N_{iter} being the total budget of iterations, so that every set of parameters is applied for a minimum number of steps to achieve improvement. In the following we denote by $x^{(i)}_{\max}$ the point attaining the highest loss found until iteration i, and by $M = \{n \in \mathbb{N} \,|\, n \mod m = 0\}$ the set of iterations at which the parameters are recomputed.

Selection of sparsity: We choose an update step for l_1-APGD whose sparsity is automatically computed by considering the best point found so far. In order to have both

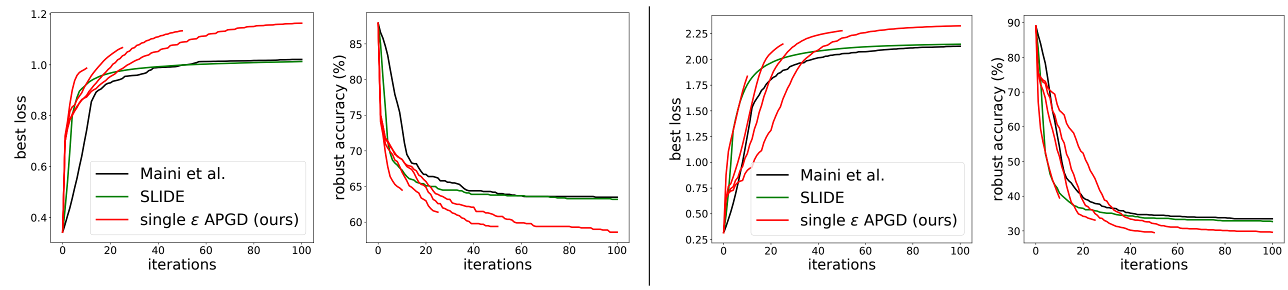

Figure 3. Plots of best robust loss obtained so far (first/third) and robust accuracy (second/fourth) over iterations for the l_1-PGD of (Tramèr & Boneh, 2019) (SLIDE), the one of (Maini et al., 2020) and our single-ϵ l_1-APGD with 10 (the version used for adversarial training), 25, 50 and 100 iterations for two models (left: our own l_1-robust model, right: the l_2-robust one of (Rice et al., 2020)). Since our method relies on an adaptive scheme, it automatically adjusts the parameters to the number of available iterations, outperforming the competitors.

sufficiently fast improvements and a good exploration of the feasible set in the first iterations of the algorithm, we start with updates $d/5$ with nonzero elements, that is a sparsity $k^{(0)} = 0.2$ (in practice this implies $k^{(0)} \gg 2\epsilon/d$). Then, the sparsity of the updates is adjusted as

$$k^{(i)} = \begin{cases} \left\| x_{\text{max}}^{(i-1)} - x \right\|_0 / (1.5 \cdot d) & \text{if } i \in M, \\ k^{(i-1)} & \text{else.} \end{cases} \quad (9)$$

Note that $k^{(i)}$ is smaller than the sparsity of the current best perturbation since the initial perturbations have much larger l_0-norm than the expected 2ϵ, and we want to refine them.

Selection of the step size: Simultaneously to the sparsity of the updates, we adapt the step size to the trend of the optimization. As high level idea: if the l_0-norm of the best solution is not decreasing significantly for many iterations, this suggests that it is close to the optimal value and that the step size is too large to make progress, then we reduce it. Conversely, when the sparsity of the updates keeps increasing we want to allow large step sizes since the region of the feasible set which can be explored by sparser updates is different from what can be seen with less sparse steps. We set the initial step size $\eta^{(0)} = \epsilon$ (the radius of the l_1-ball), so that the algorithm can search efficiently the feasible set, and then adjust the step size at iterations $i \in M$ according to

$$\eta^{(i)} = \begin{cases} \max\{\eta^{(i-m)}/1.5, \eta_{\text{min}}\} & \text{if } k^{(i)}/k^{(i-m)} \geq 0.95, \\ \eta^{(0)} & \text{else,} \end{cases} \quad (10)$$

where $\eta_{\text{min}} = \epsilon/10$ is the smallest value we allow for the step size. Finally, when the step size is set to its highest value, the algorithm restarts from $x_{\text{max}}^{(i)}$.

3.2. Multi-ϵ l_1-APGD

In the described l_1-APGD all iterates belong to the feasible set S. However, since the points maximizing the loss are most likely on the low dimensional faces of S, finding them might require many iterations. We notice that the same points are instead in the interior of any l_1-ball with radius larger than ϵ. Thus we propose to split N_{iter} into three phases with 30%, 30% and 40% of the iteration budget, where we optimize the objective L in l_1-balls of radii 3ϵ, 2ϵ and ϵ (always intersected with $[0, 1]^d$) respectively. At the beginning of each phase the output of the previous one is projected onto the intersection of the next l_1-ball and $[0, 1]^d$ and used as starting point for l_1-APGD with smaller radius. In this way we efficiently find good regions where to start the optimization in the target feasible set, see Figure 2 for an illustration.

3.3. Comparison PGD vs single-ϵ and multi-ϵ l_1-APGD

In Figure 2 we compare the performance of the versions of PGD used by SLIDE (Tramèr & Boneh, 2019) and (Maini et al., 2020) to our l_1-APGD, in both single- and multi-ϵ variants. For two models on CIFAR-10, we plot the best average (over 1000 test points) cross-entropy loss achieved so far and the relative robust accuracy (classification accuracy on the adversarial points), with a total budget of 100 iterations. For multi-ϵ APGD, we report the results only from iteration 60 onward, since before that point the iterates are outside the feasible set and thus the statistics not comparable. We see that the single-ϵ APGD achieves higher (better) loss and lower (better) robust accuracy than the existing PGD-based attacks, which tend to quickly plateau. Also, multi-ϵ APGD provides an additional improvement: exploring the larger l_1-ball yields an initialization in S with high loss, from where even a few optimization steps are sufficient to outperform the other methods. Additionally, we observe that using the exact projection (solid lines) boosts the effectiveness of the attacks compared to the approximated one (dashed lines), especially on the l_2-robust model of (Rice et al., 2020) (the two rightmost plots). Moreover, we show in Figure 3 how our single-ϵ APGD adapts to different budgets of iterations: when more steps are available, the loss improves more slowly at the beginning, favoring the exploration of the feasible set, but it finally achieves better values. Note that in this way, even with only 25 steps, l_1-APGD outperforms existing methods with 100 iterations both in terms of loss and robust accuracy attained.

3.4. Adversarial training with l_1-APGD

A natural application of a strong PGD-based attack is to maximize the loss in the inner maximization problem of adversarial training (AT) (Madry et al., 2018) and finding points attaining higher loss should lead to more adversarially robust classifiers. Prior works have shown that performing adversarial training wrt l_1 is a more delicate task than for other l_p-threat models: for CIFAR-10, (Maini et al., 2020) report that using PGD wrt l_1 in AT led to severe gradient obfuscation, and the resulting model is less than 8% robust at $\epsilon = 12$. A similar effect is reported in (Liu et al., 2020), where the B&B attack of (Brendel et al., 2019) more than halves the robust accuracy computed by l_1-PGD used for their l_1-AT model. Both report the highest robustness to l_1-attacks when training for simultaneous robustness against different l_p-norms. In our own experiment, which are discussed in more details in App. B, we observed that AT wrt l_1, even with multi-step PGD, is prone to catastrophic overfitting (CO) as described by (Wong et al., 2020). Thus, even if the AT training seemingly works, the resulting classifier is still non-robust, suggesting some kind of overfitting to the adversarial samples generated by l_1-PGD. While we have no final explanation for this, our current hypothesis is that standard l_1-PGD produces adversarial samples of a certain sparsity level with little variation and thus the full threat model is not explored during training. In contrast, l_1-APGD, which progressively and adaptively adjusts the sparsity, mitigates the risk of CO while providing strong adversarial perturbations. We leave it to future work to do a more thorough investigation of this interesting phenomenon. We apply l_1-APGD (single-ϵ formulation with initial sparsity $k^{(0)} = 0.05$) with 10 steps to train a ResNet-18 (details in App. D) for an l_1-threat model with radius $\epsilon = 12$. In Sec. 5 we show that our APGD-AT model achieves significantly higher robust accuracy than the currently best model, even in the worst case evaluation over many strong attacks, including black-box ones.

4. l_1-AutoAttack

(Croce & Hein, 2020b) propose AutoAttack (AA), an ensemble of four diverse attacks for a standardized parameter-free and reliable evaluation of robustness against l_∞- and l_2-type attacks, and we aim to extend this framework to the case of l_1-robustness. AA includes the l_∞- and l_2-APGD optimizing either the cross-entropy (CE) or targeted version of the difference of logits ratio (T-DLR) loss (Croce & Hein, 2020b): analogously we use our multi-ϵ l_1-APGD with 5 runs (with random restarts) of 100 iterations for the CE and the T-DLR loss (total budget of 1000 steps). The targeted FAB-attack included in AA (Croce & Hein, 2020a) minimizes the norm of the adversarial perturbations and has an l_1-version, therefore no action is needed (run with top 9 classes). The black-box Square Attack (Andriushchenko et al., 2020) has only versions for l_∞- and l_2-bounded perturbations, hence we adapt the latter to the $l_1 \cap [0, 1]^d$-threat model (details in Sec. 4.1). We show in the experiments that having a black-box method helps to accurately estimate robustness even in presence of defenses with gradient obfuscation. For both FAB^T and Square Attack (5000 queries) we keep the budget of iterations and restarts defined in AA for l_∞ and l_2. Note that the parameters of all attacks are fixed so that no tuning is necessary when testing different models and thus we get a parameter-free l_1-AutoAttack which achieves SOTA performance as we show in Section 5.

4.1. l_1-Square Attack

(Andriushchenko et al., 2020) introduce Square Attack, a query efficient score-based black-box adversarial attack for l_∞- and l_2-bounded perturbations. It is based on random search and does not rely on any gradient estimation technique. (Andriushchenko et al., 2020) show that it does not suffer from gradient masking and is even competitive with white-box attacks in some scenarios. We adapt its l_2 version to our l_1-threat model, by modifying Algorithm 3 in the original paper so that all normalization operations are computed wrt the l_1-norm (see App. C for details). While (Andriushchenko et al., 2020) create at every iteration perturbations on the surface of the l_p-ball and then clip them to $[0, 1]^d$, this results in poor performance for the l_1-ball, likely due to the complex structure of the intersection of l_1-ball and $[0, 1]^d$ (see discussion above). Thus, at each iteration, we upscale the square-shaped candidate update by a factor of 3 and then project the resulting iterate onto the intersection of the l_1-ball and $[0, 1]^d$ and accept this update if it increases the loss. This procedure increases the sparsity of the iterates and in turn the effectiveness of the attack, showing again the different role that the box has in the l_1-threat model compared to the l_∞- and l_2-threat models. We show below that our resulting scheme, l_1-Square Attack, outperforms the existing black-box methods (Schott et al., 2019; Zhao et al., 2019) on a variety of models, often with margin.

5. Experiments

In the following we test the effectiveness of our proposed attacks.[1] First we compare l_1-APGD (with CE loss) and our

[1] Code available at https://github.com/fra31/auto-attack.

Table 1. **Low Budget ($\epsilon = 12$):** Robust accuracy achieved by the SOTA l_1 -adversarial attacks on various models for CIFAR-10 in the l_1-threat model with radius $\epsilon = 12$ of the l_1-ball. The statistics are computed on 1000 points of the test set. PA and Square are black-box attacks. The budget is 100 iterations for white-box attacks ($\times 9$ for EAD and +10 for B&B) and 5000 queries for our l_1-Square-Attack.

model	clean	EAD	ALMA	SLIDE	B&B	FAB^{T}	APGD$_{CE}$	PA	Square
APGD-AT **(ours)**	87.1	64.6	65.0	66.6	62.4	67.5	**61.3**	79.7	71.8
(Madaan et al., 2021)	82.0	55.3	58.1	56.1	55.2	56.8	**54.7**	73.1	62.8
(Maini et al., 2020) - AVG	84.6	51.8	54.2	53.8	52.1	61.8	**50.4**	77.4	68.4
(Maini et al., 2020) - MSD	82.1	51.6	55.4	53.2	50.7	54.6	**49.7**	72.7	63.5
(Augustin et al., 2020)	91.1	48.9	50.7	48.8	42.1	50.4	**37.1**	73.2	56.8
(Engstrom et al., 2019) - l_2	91.5	40.3	46.4	35.1	36.8	39.9	**30.2**	71.7	52.7
(Rice et al., 2020)	89.1	37.7	45.2	32.3	35.2	37.0	**27.1**	70.5	50.3
(Xiao et al., 2020)	79.4	44.9	74.5	33.3	72.6	78.9	41.4	36.2	**20.2**
(Kim et al., 2020)*	81.9	26.7	31.8	25.1	23.8	32.4	**18.9**	54.9	36.0
(Carmon et al., 2019)	90.3	25.1	18.4	19.7	18.7	31.1	**13.1**	60.8	34.5
(Xu & Yang, 2020)	83.8	20.1	24.0	18.2	14.7	27.8	**10.9**	57.0	32.0
(Engstrom et al., 2019) - l_∞	88.7	14.5	19.4	14.2	12.2	20.9	**8.0**	57.6	28.0

Table 2. **High Budget ($\epsilon = 12$):** Robust accuracy achieved by the SOTA l_1 -adversarial attacks on various models for CIFAR-10 in the l_1-threat model with l_1-radius of $\epsilon = 12$. The statistics are computed on 1000 points of the test set. "WC" denotes the pointwise worst-case over all restarts/runs of EAD, ALMA, SLIDE, B&B and Pointwise Attack. Note that APGD$_{CE+T}$, the combination of APGD$_{CE}$ and APGD$_{T\text{-}DLR}$ (5 restarts each), yields a similar performance as AA (ensemble of APGD$_{CE+T}$, l_1-FAB^T and l_1-Square Attack) with the same or smaller budget than the other individual attacks. AA performs the same or beats the worst case WC of five SOTA l_1-attacks in 8 out of 12 cases. "rep." denotes the reported robust accuracy in the original papers. * the models of (Kim et al., 2020) were not available on request and thus are retrained with their code (see appendix). ** In (Madaan et al., 2021) evaluation is done at $\epsilon = \frac{2000}{255}$, but by personal communication with the authors we found that the reported 55.0% corresponds to $\epsilon = 12$.

model	clean	EAD	ALMA	SLIDE	B&B	APGD$_{CE+T}$	WC	AA	rep.
APGD-AT **(ours)**	87.1	63.3	61.4	65.9	59.9	60.3	**59.7**	60.3	-
(Madaan et al., 2021)	82.0	54.5	54.3	55.1	51.9	51.9	**51.8**	51.9	55.0**
(Maini et al., 2020) - AVG	84.6	50.0	49.7	52.3	49.0	**46.8**	47.3	**46.8**	54.0
(Maini et al., 2020) - MSD	82.1	50.1	49.8	51.7	47.7	**46.5**	46.8	**46.5**	53.0
(Augustin et al., 2020)	91.1	46.0	42.9	41.5	32.9	31.1	31.9	**31.0**	-
(Engstrom et al., 2019) - l_2	91.5	36.4	34.7	30.6	27.5	27.0	27.1	**26.9**	-
(Rice et al., 2020)	89.1	33.9	32.4	28.1	24.2	24.2	**23.7**	24.0	-
(Xiao et al., 2020)	79.4	34.4	75.0	22.5	59.3	27.2	20.2	**16.9**	-
(Kim et al., 2020)*	81.9	24.4	22.9	19.9	15.7	15.4	**15.1**	**15.1**	81.18
(Carmon et al., 2019)	90.3	26.2	13.6	13.6	10.4	**8.3**	8.5	**8.3**	-
(Xu & Yang, 2020)	83.8	18.1	14.5	13.9	7.8	7.7	**6.9**	7.6	59.63
(Engstrom et al., 2019) - l_∞	88.7	12.5	10.0	8.7	5.9	**4.9**	5.1	**4.9**	-

l_1-Square Attack to existing white- and black-box attacks in the low budget regime, then we show that l_1-AutoAttack accurately evaluates of robustness wrt $l_1 \cap [0,1]^d$ for all the models considered. All attacks are evaluated on models trained on CIFAR-10 (Krizhevsky et al.) and we report robust accuracy for $\epsilon = 8$ and $\epsilon = 12$ on 1000 test points.

Models: The selected models are (almost all) publicly available and are representative of different architectures and training schemes: the models of (Carmon et al., 2019; Engstrom et al., 2019; Xiao et al., 2020; Kim et al., 2020; Xu & Yang, 2020) are robust wrt l_∞, where the model of (Xiao et al., 2020) is known to be non-robust but shows heavy gradient obfuscation, those of (Augustin et al., 2020; Engstrom et al., 2019; Rice et al., 2020) wrt l_2, while those of (Maini et al., 2020; Madaan et al., 2021) are trained for simultaneous robustness wrt l_∞-, l_2- and l_1-attacks. To our knowledge no prior work has focused on training robust models for solely l_1 (see discussion in Sec. 3.4). Additionally, we include the classifier we trained with l_1-APGD integrated in the adversarial training of (Madry et al., 2018) and indicated as APGD-AT (further details in the appendix).

Attacks: We compare our attacks to the existing SOTA attacks for the l_1-threat model using their existing code (see appendix for hyperparameters). In detail, we consider SLIDE (Tramèr & Boneh, 2019) (an attack based on PGD), EAD (Chen et al., 2018), FAB^{T} (Croce & Hein, 2020a), B&B (Brendel et al., 2019) and the recent ALMA (Rony et al., 2020). As reported in (Rony et al., 2020) B&B crashes as the initial procedure to sample uniform noise to get a decision different from the true class fails. Thus we initialize

Table 3. **High Budget ($\epsilon = 8$):** see Table 2 for details, the only change is the evaluation of robust accuracy at the smaller value $\epsilon = 8$.

model	clean	EAD	ALMA	SLIDE	B&B	$APGD_{CE+T}$	WC	AA
APGD-AT **(ours)**	87.1	71.6	71.8	72.9	**70.6**	**70.6**	**70.6**	**70.6**
(Madaan et al., 2021)	82.0	62.6	63.3	62.7	**60.6**	60.7	**60.6**	**60.6**
(Maini et al., 2020) - AVG	84.6	62.9	63.1	63.1	62.4	**60.3**	61.3	**60.3**
(Maini et al., 2020) - MSD	82.1	60.2	61.0	61.6	58.6	58.3	**58.2**	**58.2**
(Augustin et al., 2020)	91.1	60.9	60.1	60.8	52.6	**50.7**	52.0	**50.7**
(Engstrom et al., 2019) - l_2	91.5	54.0	54.9	52.3	46.0	44.4	45.9	**44.2**
(Rice et al., 2020)	89.1	52.5	51.5	49.9	44.3	**42.9**	44.1	**42.9**
(Kim et al., 2020)*	81.9	38.7	38.9	36.6	31.8	30.4	31.4	**30.1**
(Xiao et al., 2020)	79.4	41.2	75.3	30.7	60.6	33.8	27.9	**22.4**
(Carmon et al., 2019)	90.3	37.8	29.7	28.8	25.1	**21.1**	22.6	**21.1**
(Xu & Yang, 2020)	83.8	33.1	30.2	27.6	22.6	21.4	21.6	**21.0**
(Engstrom et al., 2019) - l_∞	88.7	28.8	24.9	23.1	17.5	16.5	16.4	**16.0**

B&B with random images from CIFAR-100 (the results do not improve when starting at CIFAR-10 images). Besides white-box methods we include the black-box Pointwise Attack (PA) (Schott et al., 2019), introduced for the l_0-threat model but successfully used as l_1-attack by e.g. (Maini et al., 2020). We always use our l_1-APGD in the multi-ϵ version.

Small Budget: We compare the attacks with a limited computational budget, i.e. 100 iterations, with the exception of EAD for which we keep the default 9 binary search steps (that is 9×100 iterations), B&B which performs an initial 10 step binary search procedure (10 additional forward passes). Moreover, we add the black-box attacks Pointwise Attack and our l_1-Square Attack with 5000 queries (no restarts). Table 1 reports the robust accuracy at $\epsilon = 12$ achieved by every attack: in all but one case l_1-$APGD_{CE}$ maximizing the cross-entropy loss outperforms the competitors, in 6 out of 11 cases with a gap larger than 4% to the second best method. Note that l_1-$APGD_{CE}$ consistently achieves lower (better) robustness with a quite significant gap to the non adaptive PGD-based attack SLIDE. Note also that l_1-$APGD_{CE}$, SLIDE and ALMA are the fastest attacks for the budget of 100 iterations (see appendix for more details). The model from (Xiao et al., 2020) exemplifies the importance of testing robustness also with black-box attacks: their defense generates gradient obfuscation so that white-box attacks have difficulties to perform well (in particular ALMA, FAB^T and B&B yield a robust accuracy close to the clean one), while Square Attack is not affected and achieves the best results with a large margin. Moreover, it outperforms on all models the other black-box attack PA. This supports its inclusion in AutoAttack.

High budget: As second comparison, we give the attacks a higher budget: we use SLIDE, FAB^T and B&B with 10 random restarts of 100 iterations (the reported accuracy is then the pointwise worst case over restarts). B&B has a default value of 1000 iterations but 10 restarts with 100 iterations each yield much better results. For ALMA and EAD we use 1000 resp. 9×1000 iterations (note that EAD does a binary search) since they do not have the option of restarts. We compare these strong attacks to the combination of l_1-$APGD_{CE}$ and l_1-$APGD_{T\text{-}DLR}$ denoted as $APGD_{CE+T}$ with 100 iterations and 5 restarts each. These runs are also part of our ensemble l_1-AutoAttack introduced in Sec. 4 which includes additionally, FAB^T (100 iterations and 9 restarts as used in l_∞- and l_2-AA) and Square Attack (5000 queries). Note that $APGD_{CE+T}$ has the same or smaller budget than the other attacks and it performs very similar to the full l_1-AutoAttack (AA). In Table 2 and 3 we report the results achieved by all methods for $\epsilon = 12$ resp. $\epsilon = 8$. AA outperforms for $\epsilon = 12$ the individual competitors in all cases except one, i.e. B&B on the APGD-AT model, where however it is only 0.5% far from the best. Also, note that all the competitors have at least one case where they report a robust accuracy more than 10% worse than AA. The same also holds for the case $\epsilon = 8$. Note that for $\epsilon = 12$ $APGD_{CE+T}$ is the best single attack in 10 out of 12 cases (B&B 3, SLIDE 1) and for $\epsilon = 8$ $APGD_{CE+T}$ is the best in 9 out of 12 cases (B&B 2, SLIDE 1).

Since AA is an ensemble of methods, as a stronger baseline we additionally report the worst case robustness across all the methods not included in AA (indicated as "WC" in the tables), that is EAD, ALMA, SLIDE, B&B and Pointwise Attack. l_1-AA achieves better results in most of the cases (7/12 for $\epsilon = 12$, 9/12 for $\epsilon = 8$) even though it has less than half of the total budget of WC. AA is 0.6% worse than WC for the APGD-AT model (at $\epsilon = 12$), which is likely due to the fact that l_1-APGD is used to generate adversarial examples at training time and thus there seems to be a slight overfitting effect to this attack. However, note that standard AT with l_1-PGD has been reported to fail completely. We report the results of the individual methods of l_1-AA in App. E.4. We also list in Table 2 the l_1-robust accuracy reported in the original papers, if available. The partially large differences indicate that a standardized evaluation with AA would lead to a more reliable assessment of l_1-robustness.

Table 4. Comparison of black-box attacks in the l_1-threat model on CIFAR-10, $\epsilon = 12$. l_1-Square Attack outperforms both the pointwise attack (Schott et al., 2019) and the l_1-ZO-ADMM-Attack (Zhao et al., 2019) by large margin.

model	clean	ADMM	PA	Square
APGD-AT **(ours)**	87.1	86.3	79.7	**71.8**
(Maini et al., 2020) - AVG	84.6	81.3	77.4	**68.4**
(Maini et al., 2020) - MSD	82.1	77.5	72.7	**63.5**
(Madaan et al., 2021)	82.0	78.4	73.1	**62.8**
(Augustin et al., 2020)	91.1	88.9	73.2	**56.8**
(Engstrom et al., 2019) - l_2	91.5	89.8	71.7	**52.7**
(Rice et al., 2020)	89.1	85.9	70.5	**50.3**
(Kim et al., 2020)*	81.9	67.8	54.9	**36.0**
(Carmon et al., 2019)	90.3	64.1	60.8	**34.5**
(Xu & Yang, 2020)	83.8	66.0	57.0	**32.0**
(Engstrom et al., 2019) - l_∞	88.7	69.3	57.6	**28.0**
(Xiao et al., 2020)	79.4	78.5	36.2	**20.2**

The most robust model is APGD-AT trained with our single-ϵ APGD (see Sec. 3.4) for $\epsilon = 12$. It improves by 7.9% over the second best model (Madaan et al., 2021) (see Table 2). This highlights how effective our l_1-APGD maximizes the target loss, even with only 10 steps used in AT.

Comparison of black-box attacks: While many black-box attacks are available for the l_∞- and l_2-threat model, and even a few have recently appeared for l_0, the l_1-threat model has received less attention: in fact, (Tramèr & Boneh, 2019; Maini et al., 2020) used the Pointwise Attack, introduced for l_0, to test robustness wrt l_1. To our knowledge only (Zhao et al., 2019) have proposed l_1-ZO-ADMM, a black-box method to minimize the l_1-norm of the adversarial perturbations, although only results on MNIST are reported. Since no code is available for ZO-ADMM for l_1, we adapted the l_2 version following (Zhao et al., 2019) (see App. D). As for our l_1-Square Attack, we give to l_1-ZO-ADMM a budget of 5000 queries of the classifier. Table 4 shows the robust accuracy on 1000 test points achieved by the three black-box attacks considered on CIFAR-10 models, with $\epsilon = 12$: Square Attack outperforms the other methods on all models, with a significant gap to the second best. Note that l_1-ZO-ADMM does not consider norm-bounded attacks, but minimizes the norm of the modifications. While it is most of the time successful in finding adversarial perturbations, it cannot reduce their l_1-norm below the threshold ϵ within the given budget of queries.

Additional experiments: We include in App. E additional experiments. In particular, we study the effect of using different values k of sparsity in SLIDE (Tramèr & Boneh, 2019): the default $k = 0.01$ achieve the best results on average on CIFAR-10, but the optimal one varies across classifiers (see App. E.1). This means that using a fixed threshold is suboptimal, and further motivates our adaptive scheme implemented in l_1-APGD. Moreover, in App. E.3 we extend the evaluation to other datasets, CIFAR-100 and ImageNet-1k, with a similar setup as above: on both datasets, l_1-APGD significantly outperforms the competitors in the low budget regime, and l_1-AA achieves similar or better results than the worst-case over all competitors.

6. Conclusion

We have shown that the proper incorporation of the box constraints in l_1-PGD attacks using the correct projection and an adaptive sparsity level motivated by the derived steepest descent direction leads to consistent improvements. Moreover, our l_1-APGD is parameter-free and adaptive to the given budget. Using l_1-APGD$_{\text{CE}}$ in AT yields up to our knowledge the most robust l_1-model for $\epsilon = 12$. We hope that reliable assessment of l_1-robustness via our l_1-AutoAttack fosters research for this particularly difficult threat model. An interesting point for future work is to study how and why APGD$_{\text{CE}}$ can avoid the failure of l_1-adversarial training.

Acknowledgements

The authors acknowledge support from the German Federal Ministry of Education and Research (BMBF) through the Tübingen AI Center (FKZ: 01IS18039A), 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

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.

Augustin, M., Meinke, A., and Hein, M. Adversarial robustness on in- and out-distribution improves explainability. In *ECCV*, 2020.

Boyd, S. and Vandenberhge, L. *Convex Optimization*. Cambridge University Press, 2004.

Brendel, W., Rauber, J., Kümmerer, M., Ustyuzhaninov, I., and Bethge, M. Accurate, reliable and fast robustness evaluation. In *NeurIPS*, 2019.

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

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

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

Condat, L. Fast projection onto the simplex and the l_1 ball. *Mathematical Programming*, 158:575–585, 2016.

Croce, F. and Hein, M. Minimally distorted adversarial examples with a fast adaptive boundary attack. In *ICML*, 2020a.

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

Croce, F., Andriushchenko, M., Sehwag, V., Flammarion, N., Chiang, M., Mittal, P., and Hein, M. Robustbench: a standardized adversarial robustness benchmark. *arXiv preprint arXiv:2010.09670*, 2020.

Duchi, J., Shalev-Shwartz, S., Singer, Y., and Chandra, T. Efficient projections onto the l1-ball for learning in high dimensions. In *ICML*, 2008.

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

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. In *NeurIPS*, 2014.

Gowal, S., Qin, C., Uesato, J., Mann, T., and Kohli, P. Uncovering the limits of adversarial training against norm-bounded adversarial examples. *arXiv preprint arXiv:2010.03593v2*, 2020.

Hall, P. The distribution of means for samples of size n drawn from a population in which the variate takes values between 0 and 1, all such values being equally probable. *Biometrika*, 19:240—245, 1927.

He, K., Zhang, X., Ren, S., and Sun, J. Identity mappings in deep residual networks. In *ECCV*, 2016.

Irwin, O. On the frequency distribution of the means of samples from a population having any law of frequency with finite moments. *Biometrika*, 19:225—239, 1927.

Kim, M., Tack, J., and Hwang, S. J. Adversarial self-supervised contrastive learning. In *NeurIPS*, 2020.

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

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

Liu, A., Tang, S., Liu, X., Chen, X., Huang, L., Tu, Z., Song, D., and Tao, D. Towards defending multiple adversarial perturbations via gated batch normalization. *arXiv preprint arXiv:2012.01654v1*, 2020.

Madaan, D., Shin, J., and Hwang, S. J. Learning to generate noise for multi-attack robustness, 2021. URL https://openreview.net/forum?id=tv8n52XbO4p.

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

Maini, P., Wong, E., and Kolter, Z. Adversarial robustness against the union of multiple perturbation models. In *ICML*, 2020.

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

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

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

Rony, J. and Ben Ayed, I. Adversarial library, 2020. URL https://github.com/jeromerony/adversarial-library.

Rony, J., Granger, E., Pedersoli, M., and Ayed, I. B. Augmented Lagrangian adversarial attacks. *arXiv preprint arXiv:2011.11857*, 2020.

Schott, L., Rauber, J., Bethge, M., and Brendel, W. Towards the first adversarially robust neural network model on MNIST. In *ICLR*, 2019.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. Intriguing properties of neural networks. In *ICLR*, pp. 2503–2511, 2014.

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

Wang, J., Zhang, T., Liu, S., Chen, P.-Y., Xu, J., Fardad, M., and Li, B. Towards a unified min-max framework for adversarial exploration and robustness. *arXiv preprint arXiv:1906.03563*, 2019.

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

Wu, B., Chen, J., Cai, D., He, X., and Gu, Q. Do wider neural networks really help adversarial robustness? *arXiv preprint arXiv:2010.01279v2*, 2021.

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

Xu, C. and Yang, M. Adversarial momentum-contrastive pre-training. *arXiv preprint, arXiv:2012.13154v2*, 2020.

Zhao, P., Liu, S., Chen, P.-Y., Hoang, N., Xu, K., Kailkhura, B., and Lin, X. On the design of black-box adversarial examples by leveraging gradient-free optimization and operator splitting method. In *ICCV*, pp. 121–130, 2019.