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 $l_2$ 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 $l_\infty$- 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 $l_\infty$- 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 $l_\infty$, 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 $l_\infty$-case. All proofs can be found in App. A.
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

\[
\begin{align*}
    u^{(i+1)} &= x^{(i)} + \eta^{(i)} \cdot s(\nabla L(x^{(i)})), \\
    x^{(i+1)} &= P_S(u^{(i+1)}),
\end{align*}
\]

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 \(\epsilon\), 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 \(\epsilon\) is significantly higher for \(l_1\) (we use \(\epsilon = 12\) similar to (Maini et al., 2020)) as for \(l_2\) (standard 0.5) and \(l_{\infty}\) (standard \(\infty \sqrt{2/\epsilon}\)) 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:

\[
\begin{align*}
P_S(u) &= \arg \min_{z \in \mathbb{R}^d} \| u - z \|_2^2 \quad \text{s.th. } \| z - x \|_1 \leq \epsilon, \\
\end{align*}
\]

and

\[
\begin{align*}
P_{B_1(x,\epsilon)}(u) &= \arg \min_{z \in \mathbb{R}^d} \| u - z \|_2^2 \quad \text{s.th. } \| z - x \|_1 \leq \epsilon.
\end{align*}
\]

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

\[
z^*_i = \begin{cases} 
1 \quad &\text{for } u_i \geq x_i \text{ and } 0 \leq \lambda^*_i \leq u_i - 1 \\
u_i - \lambda^*_i \quad &\text{for } u_i \geq x_i \text{ and } u_i - 1 < \lambda^*_i \leq u_i - x_i \\
u_i + \lambda^*_i \quad &\text{for } u_i \leq x_i \text{ and } -u_i < \lambda^*_i \leq x_i - u_i \\
0 \quad &\text{for } u_i \leq x_i \text{ and } 0 \leq \lambda^*_i \leq -u_i 
\end{cases}
\]

where \(\lambda^*_i \geq 0\). With \(\gamma \in \mathbb{R}^d\) defined as

\[
\gamma_i = \max\{-x_i \cdot \text{sign}(u_i - x_i), (1-x_i) \cdot \text{sign}(u_i - x_i)\},
\]

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

\[
\sum_{i=1}^d \max\{0, \min\{|u_i - x_i| - \lambda^*_i, \gamma_i\} \} = \epsilon.
\]

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 \(\exists u_i \in [0,1] \text{ with } u_i \neq x_i\)

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)
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 $l_{\infty}$- and $l_2$-threat models (Madry et al., 2018) the steepest descent direction (Boyd & Vandenberghe, 2004) is used in PGD, that is

$$\delta^* = \arg\max_{\delta \in \mathbb{R}^d} \langle w, \delta \rangle \quad \text{s.t.} \quad ||\delta||_p \leq \epsilon,$$  

(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 $\delta^*_p = \epsilon \text{sign}(w_i)$ 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 $j = \arg\max_i |w_i|$ the dimension corresponding to the component of $w$ with largest absolute value and $B = \{e_i\}$, the standard basis of $\mathbb{R}^d$, we have $\delta^*_1 = \epsilon \text{sign}(w_j)e_j$. 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:

$$\delta^* = \arg\max_{\delta \in \mathbb{R}^d} \langle w, \delta \rangle \quad \text{s.t.} \quad ||\delta||_1 \leq \epsilon, \quad x + \delta \in [0, 1]^d.$$  

(6)

**Proposition 2.2** Let $z_i = \max\{(1 - x_i) \text{sign}(w_i), -x_i \text{sign}(w_i)\}$, $\pi$ 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

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

(7)

Proposition 2.2 shows that adding the box-constraints leads to a steepest descent direction $\delta^*$ 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 $\delta^*$ for $\epsilon \leq \frac{d-1}{2}$, together with a simple lower bound.

![Figure 2](image-url)
Proposition 2.3 Let $w \in \mathbb{R}^d$ with $w_i \neq 0$ for all $i = 1, \ldots, d$ and $x \in \mathcal{U}([0, 1]^d)$. Then it holds for any $d - 1 \geq \epsilon > 0$,
\[
\mathbb{E}[\|\delta^*\|_0] = [\epsilon + 1] + \sum_{m=\lceil \epsilon \rceil + 2}^d \sum_{k=0}^{m-1} (-1)^k \frac{(\epsilon - k)^{m-1}}{k!(m-1-k)!} \geq \frac{3\epsilon}{2} + 1.
\]

While the exact expression is hard to access, the derived lower bound $\frac{3\epsilon}{2}$ shows that the sparsity is non-trivially bounded away from 1. For a reasonable range of $\epsilon$ the expectation is numerically larger than $2\epsilon$. 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 $g = \nabla L(x(t)), t \in \mathbb{N}$ and $T(t)$ the set of indices of the $t$ largest components of $|g|$, we define the function $s$ used in (1) via
\[
h(t) = \begin{cases} 
\text{sign}(g_i) & \text{if } i \in T(t), \\
0 & \text{else}
\end{cases}
\]
\[s(g, t) = h(t)/\|h(t)\|_1,
\]
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 $\nabla f$ 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 $l_2/l_\infty$ 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 $2\epsilon$ 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 $2\epsilon$ 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 $2\epsilon$, 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-$\epsilon$ variant which increases the effectiveness, and finally discuss its use for adversarial training (Madry et al., 2018).

3.1. Single-$\epsilon$ $l_1$-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 $N_{\text{iter}}$ 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 $x_{\max}^{(i)}$ 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 con-
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-\( \epsilon \) \( l_1 \)-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.

3.2. Multi-\( \epsilon \) \( l_1 \)-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 \( \epsilon \). Thus we propose to split \( N_{\text{iter}} \) into three phases with 30\%, 30\% and 40\% of the iteration budget, where we optimize the objective \( L \) in \( l_1 \)-balls of radii \( 3\epsilon, 2\epsilon \) and \( \epsilon \) (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-\( \epsilon \) and multi-\( \epsilon \) \( l_1 \)-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-\( \epsilon \) and multi-\( \epsilon \) 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-\( \epsilon \) 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-\( \epsilon \) APGD achieves higher (better) loss and lower (better) robust accuracy than the existing PGD-based attacks, which tend to quickly plateau. Also, multi-\( \epsilon \) 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-\( \epsilon \) APGD adapts to different budgets of iterations: when more steps are available, considering the best point found so far. In order to have both 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

\[
k^{(i)} = \begin{cases} 
\frac{x^{(i-1)} - x}{\|x^{(i-1)} - x\|_0/(1.5 \cdot d)} & \text{if } i \in M, \\
\text{else} & 
\end{cases}
\]

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 \( 2\epsilon \), 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

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

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 \( x_{\text{max}} \).
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-$\epsilon$ 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 $l_\infty$- and $l_2$-type attacks, and we aim to extend this framework to the case of $l_1$-robustness. AA includes the $l_\infty$- 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-$\epsilon$ $l_1$-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 $l_\infty$- 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 and Square Attack (5000 queries) we keep the budget of iterations and restarts defined in AA for $l_\infty$ 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 $l_\infty$- 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 $l_\infty$- 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.\footnote{Code available at https://github.com/fra31/auto-attack.} First we compare $l_1$-APGD (with CE loss) and our
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$^+$ | 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   |
| (Maimi 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$_2$ | 89.1  | 50.3| 46.4 | 35.1 | 36.8| 39.9    | **30.2**    | 71.7| 52.7   |
| (Augustin et al., 2020) | 89.1  | 37.7| 45.2 | 32.3 | 35.2| 37.0    | **27.1**    | 70.5| 50.3   |
| (Chen et al., 2020) | 79.4  | 44.9| 74.5 | 33.3 | 72.6| 78.9    | **18.9**    | 54.9| 36.0   |
| (Kim et al., 2020) | 81.9  | 26.7| 31.8 | 21.5 | 23.8| 28.4    | **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) - $l_\infty$ | 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$_{L2,Mc}$ (5 restarts each), yields a similar performance as AA (ensemble of APGD$_{CE,T}$, $l_1$-FAB$^+$ 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 $\epsilon = \frac{2000}{355}$, 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 | FAB$^+$ | 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| 47.5    | **46.8**    | 46.8| 53.0| -   |
| (Augustin et al., 2020) | 91.1  | 46.0| 42.9 | 41.5 | 32.9| 31.1    | **31.1**    | 31.9| 31.0| -   |
| (Engstrom et al., 2019) - l$_2$ | 89.1  | 33.9| 32.4 | 28.1 | 24.2| 24.2    | **23.7**    | 24.0| - | -   |
| (Xiao et al., 2020) | 79.4  | 37.6| 75.4 | 22.5 | 59.3| 53.2    | **27.2**    | 27.2| 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     | **8.3**     | 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) - $l_\infty$ | 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 $l_\infty$, 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 $l_\infty$, $l_2$, 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$^+$ (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) - $l_\infty$ | 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-$\epsilon$ 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$_{TDLR}$ 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 $l_\infty$- 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) - $l_\infty$| 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-$\epsilon$ 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 $l_\infty$- 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 $\epsilon$ 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$_{CE}$ 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$_{CE}$ 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 robust-
ness on in- and out-distribution improves explainability. In *ECCV*, 2020.

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

Brendel, W., Rauber, J., Küblerer, 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 $l_1$-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 Vald, 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.
Structure of the appendix

We provide additional details and experiments which could not be included in the main part. In summary,

- Sec. A contains the proofs omitted above and experiments in support of the effect of the exact projection $P_{S_i}(u)$ compared to the approximation $A(u)$ and about the expected sparsity of the steepest descent direction,

- in Sec. B we analyse our observations about catastrophic overfitting in adversarial training wrt $l_1$.

- Sec. C describes the algorithm of our $l_1$-Square Attack,

- in Sec. D we provide the details of models and attacks used in Sec. 5.

- Sec. E contains experiments with a larger threshold, on other datasets, an ablation study on the importance of trophic overfitting in adversarial training wrt $l_1$.

Moreover, by introducing $w = \text{sign}(u - x) w'$ we get

$$\min_{w \in \mathbb{R}^d} \frac{1}{2} \sum_{i=1}^{d} (|u_i - x_i| - w_i)^2$$

s.t. $\sum_{i=1}^{d} w_i \leq \epsilon$, $w_i \geq 0$, $\text{sign}(u_i - x_i) w_i + x_i \in [0, 1]$.

The two componentwise constraints can be summarized with

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

as $u_i \in [0, \gamma_i]$. Note that if $\gamma_i = 0$ this fixes the variable $w_i = 0$ and we then remove this variable from the optimization problem. Thus wlog we assume in the following that $\gamma_i > 0$. The corresponding Lagrangian becomes

$$L(w, \alpha, \beta, \lambda_c) = \frac{1}{2} \sum_{i=1}^{d} (|u_i - x_i| - w_i)^2 + \lambda_c \left(1, w - \epsilon\right) + \langle \alpha, w - \gamma \rangle - \langle \beta, w \rangle,$$

which yields the KKT optimality conditions for the optimal primal variable $w^*$ and dual variables $\alpha^*, \beta^* , \lambda^*_c$

$$\nabla_w L_i = w^*_i - |u_i - x_i| + \lambda^*_c + \alpha^*_i - \beta^*_i = 0$$

$$\lambda^*_c (\sum_{i=1}^{d} u_i^* \epsilon) = 0$$

$$|w_i - \gamma_i| = 0, \quad \beta^*_i w_i = 0$$

$$\lambda^*_c \geq 0, \quad \alpha^*_i \geq 0, \quad \beta^*_i \geq 0$$

Thus $\beta^*_i > 0$ implies $w^*_i = 0$ and with $\gamma_i > 0$ this yields $\alpha^*_i = 0$ and thus $\beta^* = \lambda_c - |u_i - x_i|$ and we get

$$\beta^*_i = \max\{0, \lambda^*_c - |u_i - x_i|\}.$$  

On the other hand $\alpha^*_i > 0$ implies $w^*_i = \gamma_i$ and $\beta^*_i = 0$ and thus $\alpha^*_i = |u_i - x_i| - \gamma_i - \lambda^*_c$ and thus

$$\alpha^*_i = \max\{0, |u_i - x_i| - \gamma_i - \lambda^*_c\}.$$  

Thus we have in total

$$w^*_i = \begin{cases} 
\gamma_i & \text{if } |u_i - x_i| - \gamma_i - \lambda^*_c > 0 \\
0 & \text{if } \lambda^*_c - |u_i - x_i| > 0 \\
|u_i - x_i| - \lambda^*_c & \text{else}.
\end{cases}$$

which can be summarized as $w^*_i = \max\{0, \min\{|u_i - x_i| - \lambda^*_c, \gamma_i|\}$). Finally, if $\sum_{i=1}^{d} \max\{0, \min\{|u_i - x_i|, \gamma_i|\} < \epsilon$ then $\lambda^*_c = 0$ is optimal, otherwise $\lambda^*_c > 0$ and we get the solution from the KKT condition

$$\sum_{i=1}^{d} \max\{0, \min\{|u_i - x_i| - \lambda^*_c, \gamma_i|\} = \epsilon. \quad (11)$$
Noting that
\[ \phi(\lambda_c) = \max \{0, \min \{|u_i - x_i| - \lambda_c, \gamma_i\}\} \]
is a piecewise linear and monotonically decreasing function in \( \lambda_c \), the solution can be found by sorting the union of
\[ |u_i - x_i| - \gamma_i \] and \[ |u_i - x_i| \] in non-decreasing order \( \pi \) and then starting with \( \lambda_c = 0 \) and then going through the sorted list until \( \phi(\lambda_c) < \epsilon \). In this case one has identified the interval which contains the optimal solution \( \lambda_c^* \) and computes the solution with the “active” set of components
\[ \{i \mid 0 < |u_i - x_i| - \lambda_c < \gamma_i\}, \]
via Equation (11). The algorithm is provided in Algorithm 2, where the main complexity is the initial sorting step \( O(2d \log(2d)) \) and some steps in \( O(d) \) so that the total complexity is \( O(2d \log(2d)) \).

Once we transform back to the original variable of (3) we get with the form of \( \gamma \) the solution
\[ z_i^* = x_i + \text{sign}(u_i - x_i)u_i^* \]
where
\[ z_i^* = \begin{cases} 
1 & \text{for } u_i \geq x_i \text{ and } 0 \leq \lambda_c^* \leq u_i - 1 \\
u_i - \lambda_c^* & \text{for } u_i \geq x_i \text{ and } u_i - 1 < \lambda_c^* \leq u_i - x_i \\
x_i & \text{for } \lambda_c^* > |u_i - x_i| \\
u_i + \lambda_c^* & \text{for } u_i \leq x_i \text{ and } u_i - \lambda_c^* \leq x_i - u_i \\
0 & \text{for } u_i \leq x_i \text{ and } 0 \leq \lambda_c^* \leq -u_i
\end{cases} \]

In order to argue about the approximate projection \( A(u) \) we need the same analysis for the \( l_1 \)-projection which can be derived in an analogous way and the solution for the projection onto \( B_1(x, \epsilon) \) in (4) has the form:
\[ z_1 = \begin{cases} 
u_i - \lambda_1^* & \text{for } u_i \geq x_i \text{ and } \lambda_1^* \leq |u_i - x_i| \\
x_i & \text{for } \lambda_1^* > |u_i - x_i| \\
u_i + \lambda_1^* & \text{for } u_i \leq x_i \text{ and } \lambda_1^* \leq |u_i - x_i|
\end{cases} \]
where the optimal \( \lambda_1^* \geq 0 \) is equal to 0 if \( \sum_{i=1}^{d} \max \{0, |u_i - x_i|\} \leq \epsilon \) and otherwise it fulfills
\[ \sum_{i=1}^{d} \max \{0, |u_i - x_i| - \lambda_1^*\} = \epsilon. \]

The two prior versions of PGD (Tramer & Boneh, 2019; Maini et al., 2020) for the \( l_1 \)-threat model use instead of the exact projection \( P_S \) the approximation \( A : \mathbb{R}^d \to S \)
\[ A(u) = (P_H \circ P_{B_1(x, \epsilon)})(u). \]
The following proof first shows that \( A(u) \in S \). However, it turns out that the approximation \( A(u) \) “hides” parts of \( S \) due to the following property. Note that the condition is is slightly deviating from the one of Lemma 2.1 due to a corner case when \( \|u - x\|_1 = \epsilon \). Thus we know require \( \|u - x\|_1 > \epsilon \) which then implies that \( \|P_S(u) - x\|_1 = \epsilon. \)

**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
\[ \|P_S(u) - x\|_1 > \|A(u) - x\|_1. \]

**Proof.** It holds \( A(u) \in H \) but also \( A(u) \in B_1(x, \epsilon) \) as with \( z_1 := P_{B_1(x, \epsilon)}(u) \)
\[ \|z_1 - x\| \leq \epsilon, \]
and it holds with \( P_H(z) = \max \{0, \min \{z, 1\}\} \) and \( z_A := P_H(z_1) = A(u) \) that
\[ \|z_1, i - x_i| = |z_{1,i} - z_{A,i} + z_{A,i} - x_i| = |z_{1,i} - z_{A,i} + |z_{A,i} - x_i|, \]
which follows as if \( z_{1,i} \geq 1 \) then \( z_{1,i} - z_{A,i} = z_{1,i} - 1 > 0 \) and \( 1 - x_i \geq 0 \) whereas if \( z_{1,i} < 0 \) then \( z_{1,i} - z_{A,i} = z_{1,i} - 0 < 0 \) and \( 0 - x_i \leq 0 \). Thus we get
\[ \|z_1 - x\|_1 = \|z_1 - z_A\|_1 + \|z_A - x\|_1, \tag{12} \]
Thus it holds \( \|z_A - x\|_1 \leq \|z_1 - x\|_1 \leq \epsilon \) and we get \( A(u) \in H \cap B_1(x, \epsilon) \). and thus it is a feasible point of the optimization problem in (3) and by the optimality of \( z_c := P_{B_1(x, \epsilon)} \cap H(u) \) it follows
\[ \|z_c - u\|_2 \leq \|A(u) - u\|_2. \]
If \( \|z_1 - x\|_1 < \epsilon \) then \( z_1 = u \) and thus with (12) one gets \( \|z_A - x\|_1 < \epsilon \) as well. In this case \( z_A \) is the optimal projection if we just had the box constraints. However, as \( \|z_A - x\|_1 < \epsilon \) it is also feasible for (3) and thus optimal, \( z_c = z_A \). Moreover, if \( \|u - x\|_1 = \epsilon \) then \( z_1 = u \) and thus with the same argument we get \( z_c = z_A \). On the other hand if \( \|z_1 - x\|_1 = \epsilon \) and \( z_1 \in H \), then \( z_1 \) is feasible for (3) and the minimum over a larger set and thus \( z_c = z_A \).

Suppose now that \( \|u - x\|_1 > \epsilon \) and \( z_1 \notin H \). Then \( \|z_1 - u\|_1 = \epsilon \) and there exists \( \lambda_1^* > 0 \) (optimal solution of the \( l_1 \)-projection problem) such that
\[ z_A = \begin{cases} 
\min \{u_i - \lambda_1^*, 1\} & \text{for } u_i \geq x_i \text{ and } \lambda_1^* \leq |u_i - x_i| \\
x_i & \text{for } \lambda_1^* > |u_i - x_i| \\
\max \{u_i + \lambda_1^*, 0\} & \text{for } u_i \leq x_i \text{ and } \lambda_1^* \leq |u_i - x_i|
\end{cases} \]
where we have used that \( u_i - \lambda_1^* \geq x_i \) for \( u_i \geq x_i \) resp. \( u_i + \lambda_1^* \leq x_i \) for \( u_i \leq x_i \). Moreover, as \( z_1 \notin H \) this implies
that \(|z_A - x|_1 < |z_1 - x|_1 = \epsilon\). Then if \(|z_e - x|_1 = \epsilon\) (which implies \(\lambda_e^* < \lambda_1^1\)) there is nothing to prove (we get \(|z_e - x|_1 > |z_A - x|_1\) and this represents the first condition in the Lemma for strict inequality) so suppose that \(\lambda_e^* = 0\) (that is \(|z_e - x|_1 < \epsilon\)) and thus

\[
z_e = \begin{cases} 
\min\{u_i, 1\} & \text{for } u_i \geq x_i \text{ and } |u_i - x_i| \geq 0 \\
x_i & \text{for } |u_i - x_i| < 0 \\
\max\{u_i, 0\} & \text{for } u_i \leq x_i \text{ and } |u_i - x_i| \geq 0
\end{cases}
\]

Then we get

\(|z_e)_i - x_i| \geq (z_A)_i - x_i| \quad \forall i = 1, \ldots, d.

and thus \(|z_e - x|_1 \geq |z_A - x|_1\). In fact, a stronger property can be derived under an extra condition on \(u\).

As \(|z_1 - x|_1 = \epsilon\) it must hold \(|u - x|_1 \geq \epsilon\) and thus \(u \neq x\). Now suppose that \(wog \ u_i > x_i\) and \(u_i \in [0, 1]\) then \((z_A)_i = u_i - \lambda_1^1\) if \(|u_i - x_i| \leq \lambda_1^1\) or \((z_A)_i = x_i\) if \(|u_i - x_i| > \lambda_1^1\). However, in both cases we get

\((z_e)_i - x_i = u_i - x_i > \max\{u_i - x_i - \lambda_1^1, 0\} \geq (z_A)_i - x_i,

and thus \(|z_e - x|_1 > |z_A - x|_1\). If \(u_i < x_i\) and \(u_i \in [0, 1]\) then

\((z_e)_i - x_i = u_i - x_i < \min\{u_i - x_i - \lambda_1^1, 0\} \leq (z_A)_i - x_i.

and thus \(|z_e - x|_1 > |z_A - x|_1|.

\[\square\]

In Figure 4 we simulate the projections after the steepest descent step (8) (for varying level of sparsity) to get a realistic picture of the influence of the approximation \(A(u)\) versus the exact projection \(P_S(u)\). For sparse updates (less than 50) the approximation behaves quite poorly in comparison to the exact projection in the sense that the true projection is located at the boundary of the \(l_1\)-ball around the target point \(x\) whereas \(A(u)\) is located far into the interior of the \(l_1\)-ball. Note that such sparse updates are used in SLIDE (Tramèr & Boneh, 2019) and using the exact projection improves SLIDE (see Figure 2). Next we derive the form of the steepest descent step.

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

\[
\delta_{\pi_i}^* = \begin{cases} 
z_{\pi_i} \cdot \pi \text{ sign}(w_{\pi_i}) & \text{for } i < k, \\
(\epsilon - \sum_{i=1}^{k-1} \pi_i) \cdot \pi \text{ sign}(w_{\pi_k}) & \text{for } i = k, \\
0 & \text{for } i > k
\end{cases}
\]

**Proof.** We introduce the new variable \(\alpha_i := \pi \text{ sign}(w_i) \delta_i\), with the convention \(\pi \text{ sign}(t) = 0\) if \(t = 0\). Then we get the equivalent optimization problem:

\[
\max_{\alpha \in R^d} \sum_{i} |w_i| \alpha_i
\]

s.th. \(\sum_{i=1}^{d} \alpha_i \leq \epsilon\), \(\alpha_i \geq 0\),

\[-x_i \leq \pi \text{ sign}(w_i) \alpha_i \leq 1 - x_i, \quad i = 1, \ldots, d,\]

and thus

\[0 \leq \alpha_i \leq \max\{-x_i \pi \text{ sign}(w_i), (1 - x_i) \pi \text{ sign}(w_i)\}.

Given the positivity of \(\alpha\) and the upper bounds the maximum is attained when ordering \(|w_i|\) in decreasing order \(\pi\) and setting always \(\alpha_{\pi_i}\) to the upper bound until the budget \(\sum_{i=1}^{d} \alpha_i = \epsilon\) is attained. Thus with \(k\) being the smallest integer in \([1, d]\) such that \(\sum_{i=1}^{k} \alpha_i \leq \epsilon\) we get the solution

\[
\alpha_{\pi_i}^* = \begin{cases} 
z_{\pi_i} \pi \text{ sign}(w_{\pi_i}) & \text{for } i < k, \\
(\epsilon - \sum_{i=1}^{k-1} \pi_i) \cdot \pi \text{ sign}(w_{\pi_k}) & \text{for } i = k, \\
0 & \text{for } i > k
\end{cases}
\]

Noting that \(\delta_i = \pi \text{ sign}(w_i) \alpha_i\) we get

\[
\delta_{\pi_i}^* = \begin{cases} 
z_{\pi_i} \cdot \pi \text{ sign}(w_{\pi_i}) & \text{for } i < k, \\
(\epsilon - \sum_{i=1}^{k-1} \pi_i) \cdot \pi \text{ sign}(w_{\pi_k}) & \text{for } i = k, \\
0 & \text{for } i > k
\end{cases}
\]

\[\square\]
Next we show the expected sparsity of the steepest descent step.

**Proposition 2.3** Let \( w \in \mathbb{R}^d \) with \( w_i \neq 0 \) for all \( i = 1, \ldots, d \) and \( x \in \mathcal{U}([0, 1]^d) \) and let \( \delta^* \) be the solution from Proposition 2.2. Then it holds for any \( \frac{d^2}{2} \geq \epsilon > 0 \),

\[
E[\|\delta^*\|_0] = \epsilon + 1 + \sum_{m = \lceil \epsilon \rceil + 2}^{d} \sum_{k=0}^{m} (-1)^k \frac{(\epsilon - k)^{m-1}}{k! (m - 1 - k)!} \geq \frac{3\epsilon}{2} + 1.
\]

**Proof.** We first note that the components \( z_i \) defined in Proposition 2.2 are independent and have distribution \( z_i \sim \mathcal{U}([0, 1]), i = 1, \ldots, d \) as both \( 1 - x_i \) and \( x_i \) are uniformly distributed and the \( x_i \) are independent. As the \( z_i \) are i.i.d. the distribution of \( k \) is independent of the ordering of \( w \) and thus we can just consider the probability \( \sum_{i=1}^{k} z_i \geq \epsilon \). Note that for any \( \epsilon > 0 \), we have \( 1 \leq k \leq d \). Moreover, we note that

\[
k > m \iff \sum_{i=1}^{m} z_i < \epsilon \quad \text{and} \quad k = d \iff \sum_{i=1}^{d} z_i \leq \epsilon,
\]

and as \( k \) is an integer valued random variable we have

\[
k \geq m \iff k - m - 1.
\]

Thus as \( k \) is a non-negative integer valued random variable, we have

\[
E[k] = \sum_{m=1}^{d} P(k \geq m) = \sum_{m=1}^{d} P(k > m - 1)
= \sum_{m=1}^{d} P\left( \sum_{i=1}^{m-1} z_i < \epsilon \right) = \sum_{m=1}^{d} P\left( \sum_{i=1}^{m-1} z_i \leq \epsilon \right)
\]

where in the last step we use that \( \sum_{i=1}^{m-1} z_i \) is a continuous random variable and thus we add a set of measure zero. The sum of uniformly distributed random variables on \([0, 1]\) has the Irwin-Hall distribution with a cumulative distribution function (Irwin, 1927; Hall, 1927) given by

\[
P\left( \sum_{i=1}^{m-1} z_i \leq \epsilon \right) = \frac{1}{(m-1)!} \sum_{k=0}^{m-1} (-1)^k \binom{m-1}{k} (\epsilon - k)^{m-1}.
\]

Note that the distribution of \( \sum_{i=1}^{m-1} z_i \) is symmetric around the mean value \( \frac{m-1}{2} \) and thus the median is also \( \frac{m-1}{2} \). We note that for \( m = 1 \) the first sum is empty and thus

\[
P\left( \sum_{i=1}^{0} z_i \leq \epsilon \right) = 1 \quad \text{and in general as} \quad 0 \leq z_1 \leq 1 \quad \text{it holds for} \quad d - 1 \geq \epsilon \geq m - 1:
\]

\[
P\left( \sum_{i=1}^{m-1} z_i \leq \epsilon \right) = 1
\]

\[
\sum_{m=1}^{d} P\left( \sum_{i=1}^{m} z_i \leq \epsilon \right) \geq \frac{3\epsilon}{2} + 1.
\]

Thus

\[
\sum_{m=1}^{d} P\left( \sum_{i=1}^{m-1} z_i \leq \epsilon \right) \geq \frac{3\epsilon}{2} + 1 + \frac{\epsilon + 1}{2}
\]

and thus

\[
\sum_{m=1}^{d} P\left( \sum_{i=1}^{m-1} z_i \leq \epsilon \right) \geq \frac{\epsilon + 1}{2} + \frac{\epsilon + 1}{2} \geq \frac{3\epsilon}{2} + 1.
\]

\[\Box\]

Note that tighter lower bounds could be derived with more sophisticated technical tools but we decided for the simple argument as we just want to show that the sparsity is non-trivially lower bounded. The exact expression is difficult to evaluate in high dimensions. In Figure 5 we provide an empirical evaluation of the sparsity of \( \delta^* \) for \( d = 3024 \) and \( \epsilon = 12 \) (histogram computed using 100,000 samples from \( x \in \mathcal{U}([0, 1]^d) \) and \( w \in \mathcal{N}(0, 1) \)). The exact expected sparsity can be computed as \( E[\|\delta^*\|_0] \approx 24.6667 \).

Thus

\[
\sum_{m=1}^{d} P\left( \sum_{i=1}^{m-1} z_i \leq \epsilon \right) = \epsilon + 1 + \sum_{m=\lceil \epsilon \rceil + 1}^{d} P\left( \sum_{i=1}^{m-1} z_i \leq \epsilon \right).
\]

As the median is given by \( \frac{m-1}{2} \) we get for \( \epsilon \geq \frac{m-1}{2} \)

\[
P\left( \sum_{i=1}^{m-1} z_i \leq \epsilon \right) \geq \frac{1}{2}
\]

and thus

\[
\sum_{m=1}^{d} P\left( \sum_{i=1}^{m-1} z_i \leq \epsilon \right) \geq \epsilon + 1 + \frac{2\epsilon + 1}{2} - \frac{\epsilon + 1}{2} + \sum_{m=\lceil \epsilon \rceil + 1}^{d} P\left( \sum_{i=1}^{m-1} z_i \leq \epsilon \right)
\]

\[
\geq \epsilon + 1 + \frac{2\epsilon + 1}{2} \geq \frac{3\epsilon}{2} + 1
\]

\[\Box\]

**B. Adversarial training wrt \( l_1 \)**

As mentioned in Sec. 3.4, adversarial training (Madry et al., 2018) in the \( l_1 \cap [0, 1]^d \)-threat model is more delicate than for
Figure 6. **Left:** Illustration of catastrophic overfitting in $l_1$ adversarial training when using the 10-step PGD-attack SLIDE (Tramèr & Boneh, 2019) with $k = 0.01$ for training. While the model is robust against the 10-step SLIDE attack, it is completely non-robust when running our stronger $l_1$-APGD attack with 100 iterations. **Right:** Catastrophic overfitting does not happen when using 10-step $l_1$-APGD for adversarial training to train AT-APGD, even when one attacks the model with much stronger attacks at test time, see the evaluation in Table 3. In both plots we report the robust accuracy on training and test set over the 30 epochs of training.

$l_\infty$ or $l_2$. In particular, we observe that using the multi-step PGD-based attack SLIDE with the standard sparsity of the updates $k = 0.01$ (here and in the following $k$ indicates the percentage of non zero elements) leads to catastrophic overfitting when training classifiers on CIFAR-10 with $\epsilon = 12$ and 10 steps. Here we mean by catastrophic overfitting that model is robust against the attack used during training even at test time but it fails completely against a stronger attack (APGD in the left part of Figure 6). This is similar to what has been reported by (Wong et al., 2020) in the $l_\infty$-threat model using FGSM (Goodfellow et al., 2014), i.e. a single step method, in adversarial training might produce classifiers resistant to FGSM but not to a stronger PGD attack. Moreover, the robustness against PGD drops abruptly during training, on both training and test sets. In our scenario, as illustrated in Figure 6 by the left plot, a similar phenomenon happens: when we use SLIDE in adversarial training the classifier is initially robust against both SLIDE with 10 steps (blue and yellow curves) and the stronger $l_1$-APGD (green and red) with 100 iterations, but around iteration 4000 the robustness computed with $l_1$-APGD goes close to 0%, while the model is still very robust against the attack used for training (which is a 10-step attack and not a single step attack). Conversely, when $l_1$-APGD with 10 steps is used for training (right plot) the model stays robust against both $l_1$-APGD with 100 iterations and SLIDE (and other attacks as shown in Sec. 5).

We could prevent catastrophic overfitting by decreasing the sparsity in SLIDE to $k = 0.05$, but this yields poor final robustness (around 50%) compared to what we achieved using $l_1$-APGD (59.7% against multiple attacks, see Table 2), since a weaker attack is used at training time. In fact, we show in Sec. E.1 that values $k > 0.01$ in SLIDE lead to worse performance in most of the cases. We hypothesize that $l_1$-APGD, adapting the sparsity of the updates per point, first prevents the model from seeing only perturbations with similar sparsity and second is able to effectively maximize the loss, allowing adversarial training to perform. We note that even (Tramèr & Boneh, 2019) observed that using the standard fixed value of $k$ leads to overfitting, and proposed as remedies random sparsity levels and 20 steps. We instead keep the usual 10 steps, and have a scheme which adaptively chooses the sparsity rather than randomly.

C. $l_1$-Square Attack

(Andriushchenko et al., 2020) introduce the Square Attack, a black-box score-based adversarial attacks, based on random search with square-shaped updates. The key component of such scheme is using an effective distribution to sample at each iteration a new candidate update of the current best point, i.e. achieving the best loss. We adapt the algorithm proposed for $l_2$-bounded attacks and give in Alg. 3 the detailed procedure constituting the sampling distribution of our version of Square Attack for the $l_1 \cap [0,1]^d$-threat model. If the new point $x^{(i)} + \delta$ attains a lower margin loss (since in this case we aim at minimizing the difference between the logit of the correct class and the largest of the others, until a different classification is achieved), then $x^{(i+1)} = x^{(i)} + \delta$, otherwise $x^{(i+1)} = x^{(i)}$. The input $w$ of Alg. 3 controls the size of the update and is progressively reduced according to a piecewise constant schedule, in turn regulated by the only free parameter of the method $p$. 
Almost all the models we used are publicly available: the classifiers from (Engstrom et al., 2019; Carmon et al., 2019; Rice et al., 2020; Augustin et al., 2020) are provided in the library RobustBench (Croce et al., 2020). Those of (Maini et al., 2020; Xu & Yang, 2020) can be found in the official pages. Moreover, (Madaan et al., 2021; Xiao et al., 2020)

made models available via OpenReview4,5. Upon request, (Kim et al., 2020) could not give access to the original models out of privacy reasons. Therefore we trained new classifiers using the official code6 following the suggested parameters. For the models denoted in (Kim et al., 2020) by “RoCL” and “RoCL+rLE” we could reproduce both clean and robust accuracy wrt $l_\infty$ with the original evaluation code, while for “RoCL+AT+SS” we could match the robust accuracy but not the clean one (here robust accuracy is the one computed using their code). However, we used this last one in our experiments since it is the most robust one wrt $l_1$.

For APGD-AT we trained a PreAct ResNet-18 (He et al., 2016), which we denote by “APGD-AT (ours)”.

Table 5. Architecture of the models used in the experimental evaluation on CIFAR-10.

| model | architecture |
|-------|--------------|
| APGD-AT (ours) | PreAct ResNet-18 |
| (Madaan et al., 2021) | WideResNet-28-10 |
| (Maini et al., 2020) - AVG | PreAct ResNet-18 |
| (Maini et al., 2020) - MSD | PreAct ResNet-18 |
| (Augustin et al., 2020) | ResNet-50 |
| (Engstrom et al., 2019) - l2 | PreAct ResNet-18 |
| (Rice et al., 2020) | DenseNet-121 |
| (Xiao et al., 2020) | ResNet-18 |
| (Kim et al., 2020)* | WideResNet-28-10 |
| (Carmon et al., 2019) | ResNet-18 |
| (Xu & Yang, 2020) | ResNet-50 |
| (Engstrom et al., 2019) - $l_\infty$ | ResNet-50 |

---

4https://openreview.net/forum?id=tv8n52Xb04p
5https://github.com/iclrsubmission/kwta
6https://github.com/Kim-Minseon/RoCL

D. Experimental details

We here report details about the experimental setup used in Sec. 5.

D.1. Models

Almost all the models we used are publicly available: the classifiers from (Engstrom et al., 2019; Carmon et al., 2019; Rice et al., 2020; Augustin et al., 2020) are provided in the library RobustBench (Croce et al., 2020). Those of (Maini et al., 2020; Xu & Yang, 2020) can be found in the official pages. Moreover, (Madaan et al., 2021; Xiao et al., 2020)

made models available via OpenReview4,5. Upon request, (Kim et al., 2020) could not give access to the original models out of privacy reasons. Therefore we trained new classifiers using the official code6 following the suggested parameters. For the models denoted in (Kim et al., 2020) by “RoCL” and “RoCL+rLE” we could reproduce both clean and robust accuracy wrt $l_\infty$ with the original evaluation code, while for “RoCL+AT+SS” we could match the robust accuracy but not the clean one (here robust accuracy is the one computed using their code). However, we used this last one in our experiments since it is the most robust one wrt $l_1$.

For APGD-AT we trained a PreAct ResNet-18 (He et al., 2016), which we denote by “APGD-AT (ours)”.
with softplus activation function, using cyclic learning rate with maximal value 0.1 for 100 epochs, random cropping and horizontal flipping as training set augmentations. We set 10 steps of APGD for maximizing the robust loss in the standard adversarial training setup (Madry et al., 2018).

Table 5 reports the architecture of every model. As mentioned in Sec. 5, we chose such models to have different architectures, training schemes and even training data, as (Carmon et al., 2019; Augustin et al., 2020) use unlabeled data in their methods.

D.2. Attacks

In the following we report the details of the presented attacks. We use ALMA from Adversarial Library (Rony & Ben Ayed, 2020), with the default parameters for the $l_1$-threat models, in particular $\alpha = 0.5$ with 100 iterations, $\alpha = 0.9$ with 1000. EAD, B&B and Pointwise Attack (PA) are available in FoolBox (Rauber et al., 2017): for EAD we use the $l_1$ decision rule and regularization $\beta = 0.01$, for B&B we keep the default setup, while PA does not have tunable parameters. We reimplemented SLIDE following the original code, according to which we set sparsity of the updates $k = 0.01$ for CIFAR-10 and step size $\eta = 3.06$, which is obtained rescaling the one used in (Tram`er & Boneh, 2019) $\eta = 2$ for $\epsilon = 2000/255$ (see below for a study of the effect of different values of $k$). Also, since no code is available for ZO-ADMM for $l_1$, we adapted the $l_2$ version following (Zhao et al., 2019) and then optimized its parameters, using $\rho = 2$, $\gamma = 0.1$. Finally, we use $\text{FAB}^T$ as available in AutoAttack.

For $l_1$-APGD we fix the values of all parameters to those mentioned in Sec. 3.1 and Sec. 3.2. Moreover, we set $p = 0.8$ in $l_1$-Square Attack as done in AutoAttack for $l_2$ and $l_\infty$. Thus even $l_1$-AutoAttack can be used without any parameter tuning.

**Attacks runtime:** Direct comparison of runtime is not necessarily representative of the computational cost of each method since it depends on many factors including implementation and tested classifier. We gave similar budget to (almost all) attacks: for the low budget comparison (see Table 1) we use 100 iterations, which are equivalent to 100 forward and 100 backward passes for ALMA, SLIDE and $l_1$-APGD, 110 forward and 100 backward passes for B&B (because of the initial binary search), 150 forward and 100 backward passes for $\text{FAB}^T$. EAD has instead a 9 times larger budget since we keep the default 9 binary search steps. As an example, when run using a classifier on CIFAR-10 with PreAct ResNet-18 as architecture, 1000 test points, ALMA and SLIDE take around 25 s, $l_1$-APGD 27 s, $\text{FAB}^T$ 32 s, EAD 105 s, B&B 149 s.

E. Additional experiments

E.1. Effect of sparsity in SLIDE

Since the sparsity $k$ of the updates is a key parameter in SLIDE, the PGD-based attack for $l_1$ proposed in (Tram`er & Boneh, 2019), we study the effect of varying $k$ on its performance. In Table 6 we report the robust accuracy achieved by SLIDE with 5 values of $k \in \{0.001, 0.003, 0.01, 0.03, 0.1\}$ on the CIFAR-10 models used for the experiments in Sec. 5, with a single run of 100 iterations. As a reference, we also show the results of our $l_1$-APGD with the same budget (grey column). We observe that while the default value $k = 0.01$ performs best in most of the cases, for 3/12 models the lowest robust accuracy is obtained by $k = 0.03$, for 1/12 by $k = 0.001$. This means that SLIDE would require to tune the value of $k$ for each classifier to optimize its performance. Moreover, $l_1$-APGD, which conversely automatically adapts the sparsity of the updates, outperforms the best out of the 5 versions of SLIDE in 11 out of 12 cases, with the only exception of the model from (Xiao et al., 2020) which is know to present heavy gradient obfuscation and on which the black-box Square Attack achieves the best result (see Sec. 5).

E.2. Larger threshold $\epsilon$

We here test that the performance of our attacks at the larger threshold $\epsilon = 16$. In Table 7 we run all the methods, with the lower budget, on the four most robust models: one can observe that even with a larger threshold our $l_1$-APGD outperforms the competitors on all models.

E.3. Other datasets

We test the effectiveness of our proposed attacks on CIFAR-100 and ImageNet-1k, with $\epsilon = 12$ and $\epsilon = 60$ respectively, in the same setup of Sec. 5. For CIFAR-100 we use the models (PreAct ResNet-18) from (Rice et al., 2020), for ImageNet those (ResNet-50) of (Engstrom et al., 2019): in both cases one classifier is trained for $l_\infty$-robustness, the other one for $l_2$, all are publicly available\(^7\). On ImageNet, because of the different input dimension, we use $k = 0.001$ for SLIDE (after tuning it), and we do not run Pointwise Attack since it does not scale. For B&B we use random images not classified in the target class from the respective test or validation sets as starting points. We observe that on ImageNet, the gap in runtime between B&B and the faster attacks increases significantly: for example, to run 100 steps for 1000 test points B&B takes 3612 s, that is around 14 times more than $l_1$-APGD (254 s). Thus B&B

\(^7\)https://github.com/locuslab/robust_overfitting
\(^8\)https://github.com/MadryLab/robustness/tree/master/robustness
Table 6. Effect of the sparsity $k$ of the updates in SLIDE (Tramèr & Boneh, 2019), whose default value is $k = 0.01$.

| model              | $k = 0.001$ | $k = 0.003$ | $k = 0.01$ | $k = 0.03$ | $k = 0.1$ | APGD$_{CE}$ |
|--------------------|-------------|-------------|-------------|-------------|------------|-------------|
| APGD-AT (ours)     | 74.5        | 70.1        | 66.6        | 64.4        | 65.7       | 61.3        |
| (Madaan et al., 2021) | 61.9        | 58.2        | 56.1        | 57.0        | 66.2       | 54.7        |
| (Maini et al., 2020) - AVG | 71.7        | 64.4        | 53.8        | 51.8        | 64.7       | 50.4        |
| (Maini et al., 2020) - MSD | 63.6        | 58.5        | 53.2        | 51.7        | 62.2       | 49.7        |
| (Augustin et al., 2020) | 60.9        | 53.0        | 48.8        | 59.3        | 74.1       | 37.1        |
| (Engstrom et al., 2019) - $l_2$ | 49.6        | 40.0        | 35.1        | 47.4        | 67.4       | 30.2        |
| (Xiao et al., 2020)   | 28.2        | 30.1        | 33.3        | 36.1        | 45.4       | 41.4        |
| (Rice et al., 2020)   | 47.0        | 37.6        | 32.3        | 45.2        | 65.2       | 27.1        |
| (Kim et al., 2020)    | 38.4        | 30.6        | 25.1        | 34.9        | 58.3       | 18.9        |
| (Carmon et al., 2019) | 41.4        | 29.9        | 19.7        | 24.2        | 64.4       | 13.1        |
| (Xu & Yang, 2020)     | 35.3        | 24.9        | 18.2        | 21.1        | 58.2       | 10.9        |
| (Engstrom et al., 2019) - $l_\infty$ | 34.0        | 23.6        | 14.2        | 17.0        | 59.4       | 8.0         |

Table 7. Low Budget ($\epsilon = 16$): Robust accuracy achieved by the SOTA $l_1$-adversarial attacks on models for CIFAR-10 at $\epsilon = 16$.

| model              | clean | EAD | ALMA | SLIDE | B&B | FAB$^T$ | $\text{APGD}_{CE}$ | PA | Square |
|--------------------|-------|-----|------|-------|-----|---------|--------------------|----|--------|
| APGD-AT (ours)     | 87.1  | 55.2| 55.3 | 59.1  | 52.7| 59.7    | 50.6               | 78.5| 66.4   |
| (Madaan et al., 2021) | 82.0  | 46.2| 50.0 | 47.7  | 46.0| 48.2    | 45.6               | 70.4| 58.5   |
| (Maini et al., 2020) - MSD | 82.1  | 43.9| 46.2 | 44.9  | 43.2| 48.5    | 40.9               | 69.8| 57.9   |
| (Maini et al., 2020) - AVG | 84.6  | 43.7| 44.9 | 45.9  | 43.0| 55.3    | 38.9               | 75.1| 62.9   |

Table 8. Low Budget: Robust accuracy achieved by the SOTA $l_1$-adversarial attacks on various models for CIFAR-100 and ImageNet in the $l_1$-threat model with radius indicated. 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$^T$ | $\text{APGD}_{CE}$ | PA | Square |
|--------------------|-------|-----|------|-------|-----|---------|--------------------|----|--------|
| CIFAR-100 (\(\epsilon = 12\)) |       |     |      |       |     |         |                    |    |        |
| (Rice et al., 2020) - $l_2$ | 58.7  | 19.5| 24.4 | 17.7  | 19.3| 19.6    | 14.6               | 37.0| 23.8   |
| (Rice et al., 2020) - $l_\infty$ | 54.5  | 8.6 | 9.9  | 6.0   | 6.5 | 13.0    | 4.5                | 23.0| 10.6   |
| ImageNet (\(\epsilon = 60\)) |       |     |      |       |     |         |                    |    |        |
| (Engstrom et al., 2019) - $l_2$ | 56.6  | 45.6| 50.8 | 44.7  | 44.4| 44.8    | 43.6               | -   | 50.2   |
| (Engstrom et al., 2019) - $l_\infty$ | 61.9  | 11.7| 26.6 | 11.5  | 9.5 | 34.6    | 6.3                | -   | 23.9   |

Table 9. High Budget: Robust accuracy achieved by the SOTA $l_1$-adversarial attacks on various models for CIFAR-100 and ImageNet in the $l_1$-threat model with $l_1$-radius indicated. 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, if available, Pointwise Attack. Note that APGD$_{CE+T}$, the combination of APGD$_{CE}$ and APGD$_{FAB}^T$ (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.

| model              | clean | EAD | ALMA | SLIDE | B&B | $\text{APGD}_{CE+T}$ | WC | AA |
|--------------------|-------|-----|------|-------|-----|----------------------|----|----|
| CIFAR-100 (\(\epsilon = 12\)) |       |     |      |       |     |                      |    |    |
| (Rice et al., 2020) - $l_2$ | 58.7  | 18.4| 17.1 | 15.5  | 13.1| 12.1                 | 12.7| 12.1   |
| (Rice et al., 2020) - $l_\infty$ | 54.5  | 8.1 | 5.5  | 4.5   | 3.0 | 3.4                  | 2.9 | 3.1   |
| ImageNet (\(\epsilon = 60\)) |       |     |      |       |     |                      |    |    |
| (Engstrom et al., 2019) - $l_2$ | 56.6  | 44.1| 45.6 | 44.2  | 40.3| 40.5                 | 40.3| 40.5 |
| (Engstrom et al., 2019) - $l_\infty$ | 61.9  | 9.6 | 17.1 | 8.5   | 6.2 | 4.6                  | 5.8 | 4.4   |
It scales much worse to high-resolution datasets. Also, while B&B and \( l_1 \)-APGD have in principle a similar budget in terms of forward/backward passes, one could do much more restarts for \( l_1 \)-APGD in the same time as for B&B.

We report in Table 8 and Table 9 the robust accuracy given by every attack on 1000 points of test set of CIFAR-100 or validation set of ImageNet. Similarly to CIFAR-10, our \( l_1 \)-APGD achieves the best results for all models in the low budget regime (see Table 8) with a significant gap to the second best, either B&B or SLIDE. Moreover, when using higher computational budget, \( l_1 \)-AutoAttack gives the lowest robust accuracy in 2/4 cases, improving up 1.4% over WC, the pointwise worst case over all attacks not included in AA, while in the other cases it is only 0.2% worse than WC, showing that it gives a good estimation of the robustness of the models.

**E.4. Composition of \( l_1 \)-AutoAttack**

We report in Table 10 the individual performance of the 4 methods constituting \( l_1 \)-AutoAttack, recalling that each version of \( l_1 \)-APGD, with either cross-entropy or targeted DLR loss, is used with 5 runs of 100 iterations, FAB\(^T\) exploits 9 restarts of 100 iterations and \( l_1 \)-Square Attack has a budget of 5000 queries. Note that the robust accuracy given by \( l_1 \)-AutoAttack is in all cases lower than that of the best individual attack, which varies across models.
Table 10. Individual performance of the components of $l_1$-AutoAttack.

| model                  | clean | APGD$_{CE}$ | APGD$_{TDLR}$ | FAB$^T$ | Square | AA     |
|------------------------|-------|-------------|----------------|---------|--------|--------|
| **CIFAR-10 ($\epsilon = 12$)** |       |             |                |         |        |        |
| APGD-AT (ours)         | 87.1  | 60.8        | 60.8           | 65.9    | 71.8   | 60.3   |
| (Madaan et al., 2021)  | 82.0  | 54.2        | 52.0           | 54.7    | 62.8   | 51.9   |
| (Maini et al., 2020) - AVG | 84.6  | 48.9        | 47.5           | 59.5    | 68.4   | 46.8   |
| (Maini et al., 2020) - MSD | 82.1  | 48.6        | 47.4           | 53.5    | 63.5   | 46.5   |
| (Augustin et al., 2020) | 91.1  | 34.7        | 34.5           | 42.4    | 56.8   | 31.0   |
| (Engstrom et al., 2019) - $l_2$ | 91.5  | 27.9        | 29.3           | 32.9    | 52.7   | 26.9   |
| (Rice et al., 2020)    | 89.1  | 25.5        | 26.3           | 30.3    | 50.3   | 24.0   |
| (Xiao et al., 2020)    | 79.4  | 32.2        | 33.4           | 78.6    | 20.2   | 16.9   |
| (Kim et al., 2020)     | 81.9  | 17.0        | 16.9           | 22.2    | 36.0   | 15.1   |
| (Carmon et al., 2019)  | 90.3  | 9.9         | 9.9            | 21.5    | 34.5   | 8.3    |
| (Xu & Yang, 2020)      | 83.8  | 9.6         | 9.3            | 17.7    | 32.0   | 7.6    |
| (Engstrom et al., 2019) - $l_{\infty}$ | 88.7  | 6.1         | 6.7            | 13.0    | 28.0   | 4.9    |
| **CIFAR-100 ($\epsilon = 12$)** |       |             |                |         |        |        |
| (Rice et al., 2020) - $l_2$ | 58.7  | 13.4        | 13.8           | 16.4    | 23.8   | 12.1   |
| (Rice et al., 2020) - $l_{\infty}$ | 54.5  | 3.8         | 4.2            | 7.7     | 10.6   | 3.1    |
| **ImageNet ($\epsilon = 60$)** |       |             |                |         |        |        |
| (Engstrom et al., 2019) - $l_2$ | 56.6  | 42.8        | 40.8           | 43.1    | 50.2   | 40.5   |
| (Engstrom et al., 2019) - $l_{\infty}$ | 61.9  | 5.7         | 5.5            | 18.9    | 23.9   | 4.4    |