MIND THE BOX: l_1-APGD FOR SPARSE ADVERSARIAL ATTACKS ON IMAGE CLASSIFIERS

Francesco Croce
University of Tübingen

Matthias Hein
University of Tübingen

ABSTRACT

We show that 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 $[0, 1]^d$, for which we derive the steepest descent step and the exact projection. We propose an adaptive PGD highly effective with a small budget of iterations, yielding a strong attack and, in adversarial training, models with SOTA l_1-robustness. Our l_1-APGD and a novel l_1-Square Attack form l_1-AutoAttack, an ensemble of attacks which reliably assesses adversarial robustness.

1 INTRODUCTION

In the context of adversarial robustness (Szegedy et al., 2014; Kurakin et al., 2017) the community has focused mainly on l_∞- and l_2-bounded perturbations: l_1-perturbation sets are complementary as they lead to very sparse changes. While l_1-bounded attacks exist (Chen et al., 2018; Modas et al., 2019; Brendel et al., 2019; Croce & Hein, 2020a; Rony et al., 2020), projected gradient descent (PGD) attack of Madry et al. (2018) lacks an established form for the l_1-case (Tramèr & Boneh, 2019; Maini et al., 2020). Also, obtaining l_1-robust models with adversarial training has been reported to be difficult (Maini et al., 2020; Liu et al., 2020). We identify a reason why the current l_1-PGD attacks are weaker than SOTA methods in not considering that the effective threat model is the intersection of the l_1-ball and the image domain $[0, 1]^d$. We first compute the exact projection onto such set and discuss theoretically and empirically how it improves the effectiveness of PGD. Second, we derive the correct steepest descent step in l_1-ball $\cap [0, 1]^d$ which motivates our scheme with adaptive sparsity. Then, inspired by the recent Auto-PGD (APGD) (Croce & Hein, 2020b) for l_2 and l_∞, we design a novel fully adaptive parameter-free PGD, used to train the model with the highest l_1-robust accuracy. Finally, following Croce & Hein (2020b) we assemble l_1-APGD, l_1-FAB attack (Croce & Hein, 2020a) and an l_1-adaptation of 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.

2 MIND THE BOX $[0, 1]^d$ IN l_1-PGD

PGD alternates a descent step and a projection onto the feasible set S: given the current iterate $x^{(i)}$,

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

where d is the input dimension, $\eta^{(i)} > 0$ the step size, $s : \mathbb{R}^d \to \mathbb{R}^d$ 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. We denote by $B_1(x, \epsilon) := \{z \in \mathbb{R}^d \mid \|z - x\|_1 \leq \epsilon\}$ the l_1-ball around $x \in [0, 1]^d$ and $S := [0, 1]^d \cap B_1(x, \epsilon)$. The main difference to prior work is that we take explicitly into account the image constraint $[0, 1]^d$. As ϵ is significantly higher for l_1-attacks (e.g. $\epsilon = 12$ on CIFAR-10) than for l_2 (0.5) and l_∞ ($\frac{8}{255}$) the difference of the intersection with $[0, 1]^d$ to the l_p-ball alone is most prominent for the l_1-case.

Projection onto S: With $B_1(x, \epsilon)$ as above and denoting $H = [0, 1]^d$ the image box, we define

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

and denote $P_{B_1(x,\epsilon)}(u)$ the similar projection onto the l_1-ball centered in x. $P_{B_1(x,\epsilon)}(u)$ 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 (in Sec. A we provide the derivation of the solution).

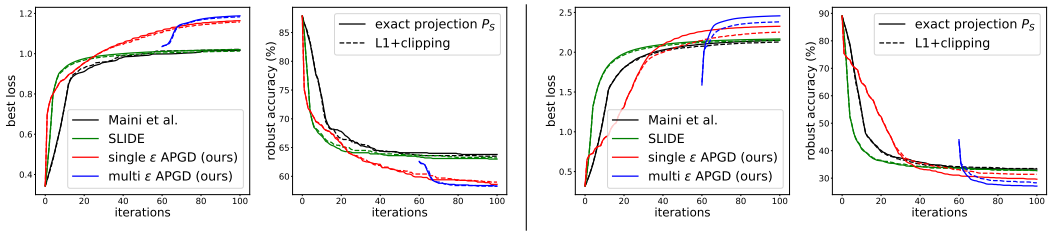

Figure 1: Best current robust loss (first/third) and robust accuracy (second/fourth) over iterations for the l_1-PGD of Tramèr & Boneh (2019) (SLIDE), the one of Maini et al. (2020) and our single ϵ-APGD and multi ϵ-APGD for two models (left: our APGD-AT, right: that of Rice et al. (2020)). All are run with either the correct projection $P_S(u)$ (solid) or the approximation $A(u)$ (dashed). The exact projection improves in almost all cases the results and APGD outperforms the competitors.

Proposition 2.1 *The projection problem* (2) *onto* $S = B_1(x, \epsilon) \cap H$ *can be solved in* $O(d \log d)$.

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. In fact, Lemma A.1 shows that the approximation $A(u)$ of $P_S(u)$ used by prior works is suboptimal under relatively weak conditions since it has a smaller l_1-distance to the target point x, i.e. $\|P_S(u) - x\|_1 > \|A(u) - x\|_1$. 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, as shown in Figure 1.

Descent direction: The next part in PGD in (1) is the choice of the descent direction $s(\nabla f(x_i))$. For l_∞- and l_2-PGD (Madry et al., 2018) the steepest descent direction (Boyd & Vandenberhge, 2004) is used, that is the vector δ_p^* which maximizes a linear function over the given l_p-ball. For $p = 1$, if $j = \arg\max_i |w_i|$ and $\mathcal{B} = \{e_i\}_i$ the standard basis of $\mathbb{R}^d$, then $\delta_1^* = \epsilon \operatorname{sign}(w_j) e_j$. Since such direction is inefficient with a small number of iterations, Tramèr & Boneh (2019) use the sign of the top-k components of the gradient (ordered according to their magnitude).

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* $\sum_{i=1}^k z_{\pi_i} \geq \epsilon$. *The steepest descent direction in* $B_1(x, \epsilon) \cap H$ *is given elementwise by*

$$\delta^*_{\pi_i} = z_{\pi_i} \operatorname{sign}(w_{\pi_i}) \text{ if } i < k, \quad \delta^*_{\pi_i} = (\epsilon - \sum_{i=1}^{k-1} z_{\pi_i}) \operatorname{sign}(w_{\pi_k}) \text{ if } i = k, \quad \delta^*_{\pi_i} = 0 \text{ else}. \tag{3}$$

Thus, 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. Proposition A.1 computes the expected sparsity of δ^* for $\epsilon \leq \frac{d-1}{2}$ and the simple lower bound $\frac{\lfloor 3\epsilon \rfloor + 1}{2}$ on it, showing that that the sparsity is non-trivially bounded away from 1 (often numerically larger than 2ϵ). This justifies the heuristic non-sparse update steps of Tramèr & Boneh (2019). 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

$$h(t)_i = \begin{cases} \operatorname{sign}(g_i) & \text{if } i \in T(t) \\ 0 & \text{else} \end{cases}, \quad s(g, t) = h(t) / \|h(t)\|_1 . \tag{4}$$

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

Similarly to APGD for l_2/l_∞ in Croce & Hein (2020b), Our l_1-APGD should be parameter-free and adapt the trade-off between exploration and local fine-tuning to the given budget of iterations N_{iter}. Moreover, Proposition 2.2 suggests that the sparsity of the steepest descent direction depends on the target point and the gradient of the loss. These motivate our scheme where 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 currently best (highest loss) perturbation. Thus, initially many coordinates are updated fostering fast progress and exploration of the feasible set, while later on we have a more local exploitation by sparser updates. We adjust the sparsity and the step size $\eta^{(i)}$ every $m = \lceil 0.04 \cdot N_{\text{iter}} \rceil$ steps. We denote by $x^{(i)}_{\text{max}}$ 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. We start with updates of sparsity $k^{(0)} = 0.2$ (in practice $k^{(0)} \gg 2\epsilon/d$), later adjusted as

$$k^{(i)} = \begin{cases} \left\| x^{(i-1)}_{\text{max}} - x \right\|_0 / (1.5 \cdot d) & \text{if } i \in M, \\ k^{(i-1)} & \text{else.} \end{cases} \tag{5}$$

If the l_0-norm of the best solution is not decreasing significantly for many iterations, this is likely close to the optimal value and the step size is too large to make progress, then we reduce it. Conversely, when the sparsity of the updates keeps increasing we allow large step sizes since the region of S explored by sparser updates is different from what seen with less sparse steps. We set the initial step size $\eta^{(0)} = \epsilon$, to search efficiently the feasible set, then adjust it at $i \in M$, with $\eta_{\text{min}} = \epsilon/10$, by

$$\eta^{(i)} = \begin{cases} \max\{\eta^{(i-m)}/1.5, \eta_{\text{min}}\} & \text{if } k^{(i)}/k^{(i-m)} \geq 0.95, \\ \eta^{(0)} & \text{else,} \end{cases} \tag{6}$$

Finally, when the step size is set to its highest value, the algorithm restarts from $x^{(i)}_{\text{max}}$.

Multi-ϵ l_1-APGD: To further improve our scheme we 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. Figure 1 shows that thus we efficiently find good starting points which improve the final results.

3.1 Adversarial training with l_1-APGD

Adversarial training (AT) (Madry et al., 2018) wrt l_1 is a more delicate task than for other l_p-threat models, as reported by Maini et al. (2020); Liu et al. (2020) both achieving the highest robustness wrt l_1 when training against different l_p-attacks simultaneously. Moreover, 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). Our current hypothesis is that standard l_1-PGD produces adversarial samples of a certain sparsity level with little variation leading to overfitting. In contrast, l_1-APGD, which progressively and adaptively adjusts the sparsity, mitigates the risk of CO while providing strong adversarial perturbations. We apply l_1-APGD (single-ϵ formulation, $k^{(0)} = 0.05$) with 10 steps to train a ResNet-18: our APGD-AT model achieves significantly higher robust accuracy than the currently best model (see Sec. 5).

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 l_1. Then we use our multi-ϵ l_1-APGD with 5 runs (with random restarts) of 100 iterations one the cross-entropy (CE) and the targeted DLR (T-DLR) loss (Croce & Hein, 2020b) (total budget of 1000 steps). The other components of AA are the targeted FAB-attack (Croce & Hein, 2020a) which has an l_1-version, and the black-box Square Attack (Andriushchenko et al., 2020) which we adapt to the $l_1 \cap [0, 1]^d$-threat model (Sec. D and Sec. F). The parameters of all attacks are fixed, getting a parameter-free l_1-AutoAttack which achieves SOTA performance (Sec. 5).

5 Experiments

In the following we test the effectiveness of our proposed attacks. All attacks are evaluated on models trained on CIFAR-10 and we report robust accuracy for $\epsilon = 12$ on 1000 test points. We compare our attacks to the SOTA attacks for the l_1-threat model, SLIDE (Tramèr & Boneh, 2019) (based on PGD), EAD (Chen et al., 2018), FAB$^{\text{T}}$ (Croce & Hein, 2020a), B&B (Brendel et al., 2019), ALMA

Table 1: **Low Budget:** Robust accuracy achieved by l_1-attacks with bound $\epsilon = 12$. PA and Square are black-box. We use 100 iterations for (most) white-box attacks, 5000 queries for our l_1-Square.

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

Table 2: **High Budget:** Robust accuracy by l_1-attacks. WC and AA are ensembles of attacks, "rep." the reported robust accuracy in the original papers. * indicates retrained models. ** Madaan et al. (2021) use $\epsilon = \frac{2000}{255}$, but by personal communication the authors confirmed that 55.0% is for $\epsilon = 12$.

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**	-

(Rony et al., 2020). and the black-box Pointwise Attack (PA) (Schott et al., 2019). We always use our l_1-APGD in the multi-ϵ version (details and additional results in appendix). We first set a limited computational budget (100 iterations for most of the methods) in Table 1: l_1-$APGD_{CE}$ consistently achieves lower (better) robustness with a quite significant gap to the non adaptive PGD-based attack SLIDE. 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 do not perform well (especially ALMA, FAB^T and B&B) while Square Attack is not affected and achieves the best results. Moreover, it outperforms on all models the other black-box attack PA.

In Table 4 we set a higher budget, using either 10 random restarts of 100 iterations (SLIDE, B&B) or a single run of 1000 iterations (EAD, ALMA). We combine l_1-$APGD_{CE}$ and l_1-$APGD_{T\text{-}DLR}$ ($APGD_{CE+T}$) with 100 iterations and 5 restarts each. These runs, with FAB^T (100 iterations, 9 restarts) and Square (5000 queries), form our ensemble l_1-AutoAttack (AA). Note that $APGD_{CE+T}$ has the same or smaller budget than the other attacks and it performs very similar to the full AA. AA outperforms the individual competitors in all cases except one, i.e. B&B on the APGD-AT model (only 0.5% far from the best). Also, all competitors have at least one case where they report a robust accuracy more than 10% worse than AA. Finally $APGD_{CE+T}$ is the best single attack in 10 out of 12 cases (B&B 3, SLIDE 1). Since AA is an ensemble of methods, we add the worst-case robustness across all methods not in AA, indicated as WC. AA achieves better results in most of the cases even though it has less than half of the total budget of WC. We also list the 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. Finally, the most robust model is APGD-AT trained with our single-ϵ APGD, improving by 7.9% over the second one, showing that our l_1-APGD effectively maximizes the target loss, even with only 10 steps.

REFERENCES

Maksym Andriushchenko, Francesco Croce, Nicolas Flammarion, and Matthias Hein. Square attack: a query-efficient black-box adversarial attack via random search. In *ECCV*, 2020.

Maximilian Augustin, Alexander Meinke, and Matthias Hein. Adversarial robustness on in- and out-distribution improves explainability. In *ECCV*, 2020.

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

Wieland Brendel, Jonas Rauber, Matthias Kümmerer, Ivan Ustyuzhaninov, and Matthias Bethge. Accurate, reliable and fast robustness evaluation. In *NeurIPS*, 2019.

Yair Carmon, Aditi Raghunathan, Ludwig Schmidt, John C Duchi, and Percy S Liang. Unlabeled data improves adversarial robustness. In *NeurIPS*, pp. 11190–11201. 2019.

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

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

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

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

Francesco Croce, Maksym Andriushchenko, Vikash Sehwag, Nicolas Flammarion, Mung Chiang, Prateek Mittal, and Matthias Hein. Robustbench: a standardized adversarial robustness benchmark. *arXiv preprint arXiv:2010.09670*, 2020.

John Duchi, Shai Shalev-Shwartz, Yoram Singer, and Tushar Chandra. Efficient projections onto the l1-ball for learning in high dimensions. In *ICML*, 2008.

Logan Engstrom, Andrew Ilyas, Hadi Salman, Shibani Santurkar, and Dimitris Tsipras. Robustness (python library), 2019. URL `https://github.com/MadryLab/robustness`.

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

Philip Hall. 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.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *ECCV*, 2016.

Oscar Irwin. 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.

Minseon Kim, Jihoon Tack, and Sung Ju Hwang. Adversarial self-supervised contrastive learning. In *NeurIPS*, 2020.

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

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

Aishan Liu, Shiyu Tang, Xianglong Liu, Xinyun Chen, Lei Huang, Zhuozhuo Tu, Dawn Song, and Dacheng Tao. Towards defending multiple adversarial perturbations via gated batch normalization. *arXiv preprint arXiv:2012.01654v1*, 2020.

Divyam Madaan, Jinwoo Shin, and Sung Ju Hwang. Learning to generate noise for multi-attack robustness, 2021. URL `https://openreview.net/forum?id=tv8n52XbO4p`.

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

Pratyush Maini, Eric Wong, and Zico Kolter. Adversarial robustness against the union of multiple perturbation models. In *ICML*, 2020.

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

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

Leslie Rice, Eric Wong, and J. Zico Kolter. Overfitting in adversarially robust deep learning. In *ICML*, 2020.

Jérôme Rony and Ismail Ben Ayed. Adversarial library, 2020. URL `https://github.com/jeromerony/adversarial-library`.

Jérôme Rony, Eric Granger, Marco Pedersoli, and Ismail Ben Ayed. Augmented Lagrangian adversarial attacks. *arXiv preprint arXiv:2011.11857*, 2020.

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

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

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

Eric Wong, Leslie Rice, and J. Zico Kolter. Fast is better than free: Revisiting adversarial training. In *ICLR*, 2020.

Chang Xiao, Peilin Zhong, and Changxi Zheng. Enhancing adversarial defense by k-winners-take-all. In *ICLR*, 2020.

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

Pu Zhao, Sijia Liu, Pin-Yu Chen, Nghia Hoang, Kaidi Xu, Bhavya Kailkhura, and Xue Lin. 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 propositions and proofs omitted above and experiments in support of the effect of the exact projection $P_S(u)$ compared to the approximation $A(u)$ and about the expected sparsity of the steepest descent direction,
- Sec. B provides details and analysis of our l_1-APGD,
- in Sec. C we analyse our observations about catastrophic overfitting in adversarial training wrt l_1,
- Sec. D describes the algorithm of our l_1-Square Attack,
- in Sec. E we provide the details of models and attacks used in Sec. 5,
- Sec. F contains experiments on the effectiveness of l_1-Square Attack, on other thresholds ϵ and datasets, an ablation study on the importance of tuning the parameter of the sparsity k of the updates in SLIDE.

A OMITTED THEORETICAL RESULTS AND PROOFS

In the following we provide the missing proofs from the main paper. For the convenience of the reader we state the propositions and lemmas again.

Proposition 2.1 *The projection problem* (2) *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*

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

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*

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

Proof. By introducing the variable $w' = z - x$ we transform (2) into

$$\max_{w' \in \mathbb{R}^d} \frac{1}{2} \|u - x - w'\|_2^2 \quad \text{s.th.} \quad \|w'\|_1 \leq \epsilon, \quad w' + x \in [0, 1]^d.$$

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

$$\max_{w \in \mathbb{R}^d} \frac{1}{2} \sum_{i=1}^d (|u_i - x_i| - w_i)^2$$
$$\text{s.th.} \quad \sum_{i=1}^d w_i \leq \epsilon, \quad w_i \geq 0, \quad \text{sign}(u_i - x_i)w_i + x_i \in [0, 1].$$

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 $w_i \in [0, \gamma_i]$. 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

$$\begin{aligned} L(w, \alpha, \beta, \lambda_e) =& \frac{1}{2} \sum_{i=1}^{d} (|u_i - x_i| - w_i)^2 + \lambda_e(\langle \mathbf{1}, w \rangle - \epsilon) \\ &+ \langle \alpha, w - \gamma \rangle - \langle \beta, w \rangle \,, \end{aligned}$$

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

$$\begin{aligned} \nabla_w L_i = w_i^* - |u_i - x_i| + \lambda_e^* + \alpha_i^* - \beta_i^* = 0 \\ \lambda_e^* (\sum_{i=1}^{d} w_i^* - \epsilon) = 0 \\ \alpha_i^*(w_i - \gamma_i) = 0, \qquad \beta_i^* w_i = 0 \\ \lambda_e^* \geq 0, \ \alpha_i^* \geq 0, \ \beta_i^* \geq 0 \end{aligned}$$

Thus $\beta_i^* > 0$ implies $w_i^* = 0$ and with $\gamma_i > 0$ this yields $\alpha_i^* = 0$ and thus $\beta^* = \lambda_e - |u_i - x_i|$ and we get

$$\beta_i^* = \max\{0, \lambda_e^* - |u_i - x_i|\}.$$

On the other hand $\alpha_i^* > 0$ implies $w_i^* = \gamma_i$ and $\beta_i^* = 0$ and thus $\alpha_i^* = |u_i - x_i| - \gamma_i - \lambda_e^*$ and thus

$$\alpha_i^* = \max\{0, |u_i - x_i| - \gamma_i - \lambda_e^*\}.$$

Thus we have in total

$$w_i^* = \begin{cases} \gamma_i & \text{if } |u_i - x_i| - \gamma_i - \lambda_e^* > 0 \\ 0 & \text{if } \lambda_e^* - |u_i - x_i| > 0 \\ |u_i - x_i| - \lambda_e^* & \text{else.} \end{cases}$$

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

$$\sum_{i=1}^{d} \max\{0, \min\{|u_i - x_i| - \lambda_e^*, \gamma_i\}\} = \epsilon. \tag{7}$$

Noting that

$$\phi(\lambda_e) = \max\{0, \min\{|u_i - x_i| - \lambda_e, \gamma_i\}\}$$

is a piecewise linear and monotonically decreasing function in λ_e, 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 π and then starting with $\lambda_e = 0$ and then going through the sorted list until $\phi(\lambda_e) < \epsilon$. In this case one has identified the interval which contains the optimal solution λ_e^* and computes the solution with the "active" set of components

$$\{i \,|\, 0 < |u_i - x_i| - \lambda_e < \gamma_i\},$$

via Equation (7). The algorithm is provided in Algorithm 1, 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 (2) we get with the form of γ the solution

$$\begin{aligned} z_i^* &= x_i + \operatorname{sign}(u_i - x_i) w_i^* \\ &= \begin{cases} 1 & \text{for } u_i \geq x_i \text{ and } 0 \leq \lambda_e^* \leq u_i - 1 \\ u_i - \lambda_e^* & \text{for } u_i \geq x_i \text{ and } u_i - 1 < \lambda_e^* \leq u_i - x_i \\ x_i & \text{for } \lambda_e^* > |u_i - x_i| \\ u_i + \lambda_e^* & \text{for } u_i \leq x_i \text{ and } -u_i < \lambda_e^* \leq x_i - u_i \\ 0 & \text{for } u_i \leq x_i \text{ and } 0 \leq \lambda_e^* \leq -u_i \end{cases} \end{aligned}$$

□

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 (**??**) 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 Tramèr & 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 A.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 A.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 $z_{1,i} > 1$ 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

$$\|z_1 - x_1\|_1 = \|z_1 - z_A\|_1 + \|z_A - x\|_1 , \tag{8}$$

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 (2) and by the optimality of $z_e := P_{B_1(x,\epsilon)\cap H}(u)$ it follows

$$\|z_e - u\|_2 \leq \|A(u) - u\|_2 .$$

If $\|z_1 - x\|_1 < \epsilon$ then $z_1 = u$ and thus with (8) 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 (2) and thus optimal, $z_e = z_A$. Moreover, if $\|u - x\|_1 = \epsilon$ then $z_1 = u$ and thus with the same argument we get $z_e = z_A$. On the other hand if $\|z_1 - x\|_1 = \epsilon$ and $z_1 \in H$, then z_1 is feasible for (2) and the minimum over a larger set and thus $z_e = z_1 = z_A$.

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

$$z_A = \begin{cases} \min\{u_i - \lambda_1^*, 1\} & \text{for } u_i \geq x_i \text{ and } \lambda_1^* \leq |u_i - x_i| \\ x_i & \text{for } \lambda_1^* > |u_i - x_i| \\ \max\{u_i + \lambda_1^*, 0\} & \text{for } u_i \leq x_i \text{ and } \lambda_1^* \leq |u_i - x_i| \end{cases},$$

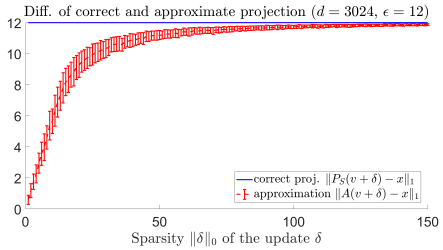

Figure 2: In order to simulate update steps of PGD, we sample target points $x \sim \mathcal{U}([0,1]^d)$ and $w \in \mathcal{N}(0,1)$ and define $v = \mathcal{P}_S(w)$ and project the point $u = v + \epsilon\delta$, where δ is generated as the descent step in (4) for different sparsity levels k. Then we plot $\|P_S(u) - x\|_1$ and $\|A(u) - x\|_1$ for varying sparsity k of the update (for each level k we show average and standard deviation over 100 samples). It can clearly be seen that the approximate projection is highly biased towards interior regions of the set $S = B_1(x, \epsilon) \cap [0,1]^d$ in particular for small levels of k whereas the correct projection stays on the surface of S. Note that SLIDE uses $k = 31$ (corresponds to 99% quantile) and thus is negatively affected by this strong bias as effectively large portions of the threat model S cannot be easily explored by SLIDE.

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 $\lambda_e^* \leq \lambda_1^*$) 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 wlog $u_i > x_i$ and $u_i \in [0,1]$ then $(z_A)_i = u_i - \lambda_1^*$ if $|u_i - x_i| \geq \lambda_1^*$ or $(z_A)_i = x_i$ if $|u_i - x_i| < \lambda_1^*$. However, in both cases we get

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

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

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

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

□

In Figure 2 we simulate the projections after the steepest descent step (4) (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 1). Next we derive the form of the steepest descent step, that is the solution of

$$\delta^* = \arg\max_{\delta \in \mathbb{R}^d} \langle w, \delta \rangle \quad \text{s.th.} \quad \|\delta\|_1 \leq \epsilon, \quad x + \delta \in [0,1]^d. \tag{9}$$

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* (9) *is given*

by

$$\delta^*_{\pi_i} = \begin{cases} z_{\pi_i} \cdot \operatorname{sign}(w_{\pi_i}) & \text{for } i < k, \\ (\epsilon - \sum_{i=1}^{k-1} z_{\pi_i}) \cdot \operatorname{sign}(w_{\pi_k}) & \text{for } i = k, . \\ 0 & \text{for } i > k \end{cases} \tag{10}$$

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

$$\begin{aligned} &\max_{\alpha \in \mathbb{R}^d_+} \ \sum_i |w_i| \alpha_i \\ &\text{s.th.} \ \sum_{i=1}^d \alpha_i \leq \epsilon, \quad \alpha_i \geq 0, \quad -x_i \leq \operatorname{sign}(w_i)\alpha_i \leq 1 - x_i, \ i = 1, \dots, d. \end{aligned} \tag{11}$$

and thus

$$0 \leq \alpha_i \leq \max\{-x_i \operatorname{sign}(w_i), (1 - x_i)\operatorname{sign}(w_i)\}.$$

Given the positivity of α and the upper bounds the maximum is attained when ordering $|w_i|$ in decreasing order π and setting always α_{π_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 $\sum_{i=1}^k u_i \leq \epsilon$ we get the solution

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

Noting that $\delta_i = \operatorname{sign}(w_i)\alpha_i$ we get

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

□

Next we show the expected sparsity of the steepest descent step.

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

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

Proof. We first note that the components z_i defined in Proposition 2.2 are independent and have distribution $z_i \sim \mathcal{U}([0, 1])$, $i = 1, \dots, d$ 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

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

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

$$\mathbb{E}[k] = \sum_{m=1}^d \mathrm{P}(k \geq m) = \sum_{m=1}^d \mathrm{P}\big(k > m - 1\big) = \sum_{m=1}^d \mathrm{P}\Big(\sum_{i=1}^{m-1} z_i < \epsilon\Big) = \sum_{m=1}^d \mathrm{P}\Big(\sum_{i=1}^{m-1} z_i \leq \epsilon\Big)$$

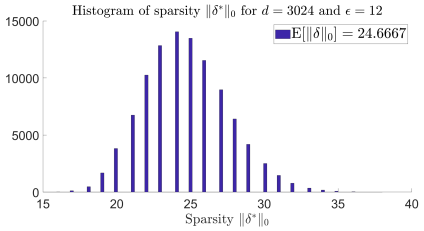

Figure 3: Histogram of the sparsity level $\|\delta^*\|_0$ of the steepest descent step from Proposition A.1 for $d = 3024$ and $\epsilon = 12$ (histogram computed using 100.000 samples from $x \in \mathcal{U}([0,1]^d)$ and $w \in \mathcal{N}(0, \mathbb{I})$). The exact expected sparsity can be computed as $\mathbb{E}[\|\delta^*\|_0] \approx 24.6667$.

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

$$\mathrm{P}\Big(\sum_{i=1}^{m-1} z_i \leq \epsilon\Big) = \frac{1}{(m-1)!} \sum_{k=0}^{\lfloor \epsilon \rfloor} (-1)^k \binom{m-1}{k} (\epsilon - k)^{m-1}.$$

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 $\mathrm{P}\Big(\sum_{i=1}^{0} z_i \leq \epsilon\Big) = 1$ and in general as $0 \leq z_1 \leq 1$ it holds for $d - 1 \geq \epsilon \geq m - 1$:

$$\mathrm{P}\Big(\sum_{i=1}^{m-1} z_i \leq \epsilon\Big) = 1$$

Thus

$$\sum_{m=1}^{d} \mathrm{P}\Big(\sum_{i=1}^{m-1} z_i \leq \epsilon\Big) = \lfloor \epsilon + 1 \rfloor + \sum_{m=\lfloor \epsilon+1 \rfloor + 1}^{d} \mathrm{P}\Big(\sum_{i=1}^{m-1} z_i \leq \epsilon\Big).$$

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

$$\mathrm{P}\Big(\sum_{i=1}^{m-1} z_i \leq \epsilon\Big) \geq \frac{1}{2}$$

and thus

$$\begin{aligned}\sum_{m=1}^{d} \mathrm{P}\Big(\sum_{i=1}^{m-1} z_i \leq \epsilon\Big) \geq & \lfloor \epsilon + 1 \rfloor + \frac{\lfloor 2\epsilon + 1 \rfloor - \lfloor \epsilon + 1 \rfloor}{2} + \sum_{m=\lfloor 2\epsilon+1 \rfloor + 1}^{d} \mathrm{P}\Big(\sum_{i=1}^{m-1} z_i \leq \epsilon\Big) \\ \geq & \frac{\lfloor \epsilon + 1 \rfloor + \lfloor 2\epsilon + 1 \rfloor}{2} \geq \frac{\lfloor 3\epsilon \rfloor + 1}{2}\end{aligned}$$

□

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 3 we provide an empirical evaluation of the sparsity of δ^* for $d = 3024$ and $\epsilon = 12$ for 100.000 samples from the uniform distribution on $[0, 1]^d$ (w is drawn from a standard multivariate Gaussian). The exact expected sparsity is about 24.6667 which is slightly higher than 2ϵ. Thus 2ϵ is a reasonable guideline in practice.

B DETAILS OF l_1-APGD

We report in Alg. 2 the detailed algorithm of our l_1-APGD.

Algorithm 1 Projection onto $B_1(x, \epsilon) \cap [0, 1]^d$

Input: point to be projected u, x and radius ϵ
Output: projection z onto $B_1(x, \epsilon) \cap [0, 1]^d$
$\gamma_i = \max\{-x_i \operatorname{sign}(u_i - x_i), (1 - x_i) \operatorname{sign}(u_i - x_i)\}$, $\lambda^* = 0$
$z_i = \min\{|u_i - x_i|, \gamma_i\}$
$S = \sum_{i=1}^d z_i$
if $S > \epsilon$ **then**
  sort $t_i = \{|u_i - x_i|, |u_i - x_i| - \gamma_i\}$ in decreasing order π and memorize if it is $|u_i - x_i|$ (category 0) or $|u_i - x_i| - \gamma$ (category 1)
  $M = |\{i \,|\, z_i = |u_i - x_i| \text{ and } |u_i - x_i| > 0\}|$
  $\lambda = 0$
  for $j = 1$ **to** 2d **do**
    $\lambda_{\text{old}} = \max\{0, \lambda\}$
    $\lambda = t_{\pi_j}$
    $S = S - M(\lambda - \lambda_{\text{old}})$
    if category$(\pi_j) = 0$ **then**
      $M = M + 1$
    else
      $M = M - 1$
    end if
    if $S < \epsilon$ **then**
      if category$(\pi_j) = 0$ **then**
        $M = M - 1$
      else
        $M = M + 1$
      end if
      $S = S + M(\lambda - \lambda_{\text{old}})$
      $\lambda^* = \lambda_{\text{old}} + (S - \epsilon)/M$
      BREAK
    end if
  end for
end if
$z_i = \max\{0, \min\{|u_i - x_i| - \lambda^*, \gamma_i\}\}, \; i = 1, \dots, d$
$P_S(u)_i = x_i + \operatorname{sign}(u_i - x_i) \cdot z_i$;

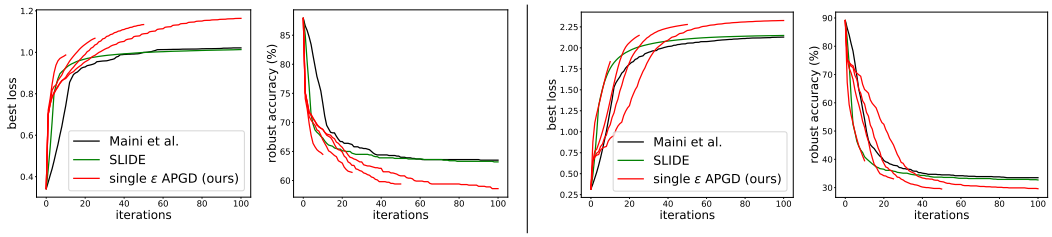

Figure 4: Plots of best robust loss obtained so far (first/third) and robust accuracy (second/fourth) over iterations for the l_1-PGD of Tramèr & Boneh (2019) (SLIDE), the one of Maini et al. (2020) and our single-ϵ l_1-APGD with 10 (the version used for adversarial training), 25, 50 and 100 iterations 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.

Moreover, we show in Figure 4 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

Algorithm 2 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. (5)
    $\eta^{(i+1)} \leftarrow$ step size as in Eq. (6)
    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

C ADVERSARIAL TRAINING WRT l_1

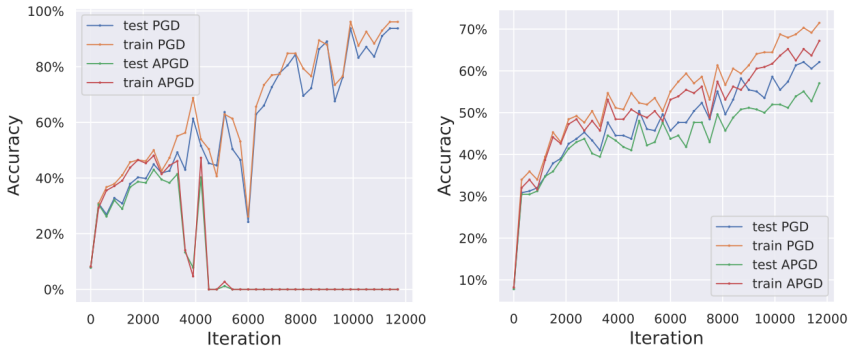

Figure 5: **Left:** Illustration of catastrophic overfitting in l_1 adversarial training when using the 10-step PGD-attack SLIDE (Tramèr & Boneh, 2019) with $k = 0.01$ for training. While the model is robust against the 10-step SLIDE attack, it is completely non-robust when running our stronger l_1-APGD 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 4. In both plots we report the robust accuracy on training and test set over the 30 epochs of training.

As mentioned in Sec. 3.1, adversarial training (Madry et al., 2018) in the $l_1 \cap [0, 1]^d$-threat model is more delicate than for l_∞ or l_2. For CIFAR-10 (Krizhevsky et al.), 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. Moreover, 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

Algorithm 3 Sampling distribution in Square Attack for $l_1 \cap [0,1]^d$

Input: target point x of shape $h \times h \times c$, radius ϵ, current iterate $x^{(i)}$, size of the windows w
Output: new update δ
$\nu \leftarrow x^{(i)} - x$
sample uniformly $r_1, s_1, r_2, s_2 \in \{0, \dots, w-h\}$
$W_1 := r_1 + 1 : r_1 + h, s_1 + 1 : s_1 + h, W_2 := r_2 + 1 : r_2 + h, s_2 + 1 : s_2 + h$
$\epsilon_{\text{unused}} \leftarrow \epsilon - \|\nu\|_1$,
$\eta^* \leftarrow \eta / \|\eta\|_1$ with η as in Eq. (2) of Andriushchenko et al. (2020)
for i = 1 **to** c **do**
  $\rho \leftarrow Uniform(\{-1, 1\})$
  $\nu_{\text{temp}} \leftarrow \rho\eta^* + \nu_{W_1,i} / \|\nu_{W_1,i}\|_1$
  $\epsilon^i_{\text{avail}} \leftarrow \|\nu_{W_1 \cup W_2,i}\|_1 + \epsilon_{\text{unused}}/c$
  $\nu_{W_2,i} \leftarrow 0, \quad \nu_{W_1,i} \leftarrow (\nu_{\text{temp}} / \|\nu_{\text{temp}}\|_1)\epsilon^i_{\text{avail}}$
end for
$z \leftarrow P_S(x + 3\nu)$
$\delta \leftarrow z - x^{(i)}$

even at test time but it fails completely against a stronger attack (APGD in the left part of Figure 5). This is similar to what has been reported by Wong et al. (2020) in the l_∞-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 5 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. F.3 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.

D l_1-SQUARE ATTACK

Andriushchenko et al. (2020) introduce the Square Attack, a query efficient black-box score-based adversarial attacks for l_∞- and l_2-bounded perturbations, based on random search with square-shaped updates. It does not rely on any gradient estimation technique, and Andriushchenko et al. (2020) show that it does not suffer from gradient masking and is even competitive with white-box attacks in some scenarios.

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 $x^{(i+1)} = x^{(i)}$. While Square Attack for l_∞ and l_2 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 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.

E EXPERIMENTAL DETAILS

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

E.1 MODELS

Table 3: 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) - l_2	ResNet-50
Rice et al. (2020)	PreAct ResNet-18
Xiao et al. (2020)	DenseNet-121
Kim et al. (2020)*	ResNet-18
Carmon et al. (2019)	WideResNet-28-10
Xu & Yang (2020)	ResNet-18
Engstrom et al. (2019) - l_∞	ResNet-50

The selected models 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. 3.1). 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.

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[1,2]. Moreover, Madaan et al. (2021); Xiao et al. (2020) made models available via OpenReview[3,4]. 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 code[5] 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_∞ 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) 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).

[1] https://github.com/locuslab/robust_union

[2] https://github.com/MTandHJ/amoc

[3] https://openreview.net/forum?id=tv8n52XbO4p

[4] https://github.com/iclrsubmission/kwta

[5] https://github.com/Kim-Minseon/RoCL

Table 3 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.

E.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. 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 B&B with random images from CIFAR-100 (the results do not improve when starting at CIFAR-10 images). 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 Tramèr & Boneh (2019) $\eta = 2$ for $\epsilon = 2000/255$ (see below for a study of the effect of different values of k). Finally, we use FAB$^{\text{T}}$ as available in AutoAttack.

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

Small budget experiments: We first 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).

High budget experiments: As second comparison, we give the attacks a higher budget: we use SLIDE, FAB$^{\text{T}}$ 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$_{\text{CE}}$ and l_1-APGD$_{\text{T-DLR}}$ denoted as APGD$_{\text{CE+T}}$ 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$^{\text{T}}$ (100 iterations and 9 restarts as used in l_∞- and l_2-AA) and Square Attack (5000 queries). Note that APGD$_{\text{CE+T}}$ has the same or smaller budget than the other attacks and it performs very similar to the full l_1-AutoAttack (AA).

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 FAB$^{\text{T}}$. 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, FAB$^{\text{T}}$ 32 s, EAD 105 s, B&B 149 s.

F ADDITIONAL EXPERIMENTS

F.1 SMALLER THRESHOLD ϵ

In Table 2 we report the performance of the various attacks when $\epsilon = 8$, that is the radius of the l_1-ball is smaller than what used in the previous experiments. Similarly to the case $\epsilon = 12$, APGD$_{\text{CE+T}}$ is the best in 9 out of 12 cases (B&B 2, SLIDE 1). Moreover, AutoAttack achieves the best robust accuracy on all the models, outperforming in most of the cases the worst-case over many other attacks (WC). Finally, even at the smaller threshold, APGD-AT yields the highest robustness.

Table 4: **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	$\text{APGD}_{\text{CE+T}}$	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**

Table 5: 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**

F.2 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) and then optimized its parameters, using $\rho = 2$, $\gamma = 0.1$. As for our l_1-Square Attack, we give to l_1-ZO-ADMM a budget of 5000 queries of the classifier. Table 5 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.

F.3 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 Tramèr & 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

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_∞	34.0	23.6	14.2	17.0	59.4	**8.0**

Table 7: **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	B&B	SLIDE	FAB^{T}	APGD_{CE}	PA	Square
CIFAR-100 ($\epsilon = 12$)									
Rice et al. (2020) - l_2	58.7	19.5	24.4	19.3	17.7	19.6	**14.6**	37.0	23.8
Rice et al. (2020) - l_∞	54.5	8.6	9.9	6.5	6.0	13.0	**4.5**	23.0	10.6
ImageNet ($\epsilon = 60$)									
Engstrom et al. (2019) - l_2	56.6	45.6	50.8	44.4	44.7	44.8	**43.6**	-	50.2
Engstrom et al. (2019) - l_∞	61.9	11.7	26.6	9.5	11.5	34.6	**6.3**	-	23.9

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).

F.4 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_∞-robustness, the other one for l_2, all are publicly available[6,7]. 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 scales much worse to high-resolution datasets. Also, while B&B and

[6] https://github.com/locuslab/robust_overfitting

[7] https://github.com/MadryLab/robustness/tree/master/robustness

Table 8: **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 $\text{APGD}_{\text{CE+T}}$, the combination of APGD_{CE} and $\text{APGD}_{\text{T-DLR}}$ (5 restarts each), yields a similar performance as AA (ensemble of $\text{APGD}_{\text{CE+T}}$, 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	$\text{APGD}_{\text{CE+T}}$	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_∞	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_∞	61.9	9.6	17.1	8.5	6.2	4.6	5.8	**4.4**

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 7 and Table 8 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 7) 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.

F.5 Composition of l_1-AutoAttack

We report in Table 9 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 9: Indivual performance of the components of l_1-AutoAttack.

model	clean	APGD_{CE}	$\text{APGD}_{\text{T-DLR}}$	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_∞	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_∞	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_∞	61.9	5.7	5.5	18.9	23.9	**4.4**