Optimization for Robustness Evaluation beyond ℓ_p Metrics

Hengyue Liang[1] LIANG656@UMN.EDU
Buyun Liang[1] LIANG664@UMN.EDU
Ying Cui[1] YINGCUI@UMN.EDU
Tim Mitchell[2] TMITCHELL@QC.CUNY.EDU
Ju Sun[1] JUSUN@UMN.EDU
[1] *University of Minnesota, Minneapolis, USA*
[2] *Queens College of the City University of New York, New York City, USA*

Abstract

Empirical evaluation of deep learning models against adversarial attacks entails solving nontrivial constrained optimization problems. Popular algorithms for solving these constrained problems rely on projected gradient descent (PGD) and require careful tuning of multiple hyperparameters. Moreover, PGD can only handle ℓ_1, ℓ_2, and ℓ_∞ attack models due to the use of analytical projectors. In this paper, we introduce a novel algorithmic framework that blends a general-purpose constrained-optimization solver `PyGRANSO`, **W**ith **C**onstraint-**F**olding (PWCF), to add reliability and generality to robustness evaluation. PWCF 1) finds good-quality solutions without the need of delicate hyperparameter tuning, and 2) can handle general attack models, e.g., general ℓ_p ($p > 0$) and perceptual attacks, which are inaccessible to PGD-based algorithms. Future updates on this topic will be posted at https://arxiv.org/abs/2210.00621.

1. Introduction

In visual recognition, deep neural networks (DNNs) are not robust against perturbations that are easily discounted by human perception—either adversarially constructed or naturally occurring [2, 10, 11, 13–15, 26, 28, 29]. A popular way of finding an adversarial perturbation (a.k.a adversarial attack) is by solving the *adversarial loss* formulation [19]:

$$\max_{\boldsymbol{x}'} \ell\big(\boldsymbol{y}, f_{\boldsymbol{\theta}}(\boldsymbol{x}')\big)\ , \quad \text{s.t. } \boldsymbol{x}' \in \Delta(\boldsymbol{x}) = \{\boldsymbol{x}' \in [0,1]^n : d\big(\boldsymbol{x}, \boldsymbol{x}'\big) \le \varepsilon\} \tag{1}$$

Here, f_θ is the DNN model, and $\Delta(\boldsymbol{x})$ is an allowable perturbation set with radius ε as measured by the metric d. Early works assume $\Delta(\boldsymbol{x})$ is the ℓ_p norm ball intersected with the natural image box, i.e., $\{\boldsymbol{x}' \in [0,1]^n : \|\boldsymbol{x} - \boldsymbol{x}'\|_p \le \varepsilon\}$, where $p = 1, 2, \infty$ are popular choices [11, 19]. To capture visually realistic perturbations, recent works have also modeled nontrivial transformations using non-ℓ_p metrics [2, 10, 13–16, 28, 29]. As for empirical robustness evaluation (RE), solutions of Eq. (1) lead to the worst-case perturbations to fool $f_{\boldsymbol{\theta}}$.

But solving Eq. (1) is not easy: the objective is non-concave for typical choices of loss ℓ and model $f_{\boldsymbol{\theta}}$; for non-ℓ_p metrics, $\Delta(\boldsymbol{x})$ is often a complicated nonconvex set. In practice, there are two major lines of algorithms: **(a) direct numerical maximization** that takes differentiable ℓ and $f_{\boldsymbol{\theta}}$, and tries direct maximization, e.g., using gradient-based methods [8, 19]. This often only produces a suboptimal solution and can lead to overoptimistic RE; **(b) upper-bound maximization** that constructs tractable upper bounds for the margin loss $\ell_{\mathrm{ML}} = \max_{i \ne y} f^i_{\boldsymbol{\theta}}(\boldsymbol{x}') - f^y_{\boldsymbol{\theta}}(\boldsymbol{x}')$, where y is the true class of $\boldsymbol{x}$, and then optimizes against the upper bounds [25]. Improving the tightness of the upper bounding while maintaining tractability is still an active area of research.

Another formalism of robustness is the *robustness radius* (or minimum distortion radius), defined as the minimal level of perturbation that causes $f_{\boldsymbol{\theta}}$ to change its predicted class:

$$\min_{\boldsymbol{x}' \in [0,1]^n} \; d(\boldsymbol{x}, \boldsymbol{x}') \quad \text{s.t. } \max_{i \neq y} f^i_{\boldsymbol{\theta}}(\boldsymbol{x}') \geq f^y_{\boldsymbol{\theta}}(\boldsymbol{x}') \tag{2}$$

Solving Eq. (2) produces not only a minimally distorted perturbation $\boldsymbol{x}'$, but also a robustness radius, which makes it another popular choice for RE [7, 8, 24]. In fact, [7, 24, 31] perform adversarial attacks by trying to solve Eq. (2).

In this paper, we focus on numerical optimization of Eq. (1). In particular, we **(I)** adapt the constrained-optimization solver `PyGRANSO` [9, 18] with a constraint-folding (PWCF) technique—crucial for making `PyGRANSO` solve Eq. (1) with reasonable speed and quality, and **(II)** show that PWCF can handle *attacks other than the ℓ_1, ℓ_2, and ℓ_∞ ones*—beyond the reach of PGD-based methods. This can lead to considerably improved RE as PWCF **(I)** can serve as a *reliable supplement* to the state-of-the-art (SOTA) RE packages on ℓ_1, ℓ_2, and ℓ_∞ attacks, e.g. `AutoAttack` [8], and **(II)** opens up the possibility of RE over a much wider range of attack models, e.g., general ℓ_p attacks with any $p > 0$ and more complicated ones such as perceptual attacks [16]. We remark that PWCF is also general enough to solve Eq. (2), but due to the limited preliminary results currently at hand, we leave it as future work.

2. Technical background

Eq. (1) is often solved by the projected gradient descent (PGD)[1] method. The basic update reads $\boldsymbol{x}'_{new} = \mathcal{P}_{\Delta(\boldsymbol{x})}(\boldsymbol{x}'_{old} + t\nabla \ell(\boldsymbol{x}'_{old}))$, where $\mathcal{P}_{\Delta(\boldsymbol{x})}$ is the projection operator onto $\Delta(\boldsymbol{x})$. When $\Delta(\boldsymbol{x}) = \{\boldsymbol{x}' \in [0,1]^n : \|\boldsymbol{x}' - \boldsymbol{x}\|_p \leq \varepsilon\}$ with $p = 1, \infty$, $\mathcal{P}_{\Delta(\boldsymbol{x})}$ takes simple forms. For $p = 2$, sequential projection onto the box and then the norm ball at least finds a feasible solution. Hence, PGD is feasible for these cases. For *other choices of* p and *general non-ℓ_p metrics* d where analytical projection is not so intuitive to derive, existing PGD based algorithms does not apply. For practical PGD methods, previous works have shown that the solution quality is sensitive to the tuning of multiple hyperparameters, e.g., step-size schedule and iteration budget [4, 8, 22]. The SOTA PGD variants, APGD-CE and APGD-DLR, try to make the tuning automatic by combining a heuristic adaptive step-size schedule and momentum acceleration under fixed iteration budget [8]—both are built into the popular `AutoAttack` package[2].

2.1. `PyGRANSO` for constrained optimization

In principle, as an instance of nonlinear optimization (NO) problems [1]

$$\min_{\boldsymbol{x}} \; g(\boldsymbol{x}) \, , \quad \text{s.t. } c_i(\boldsymbol{x}) \leq 0 \; \forall \, i \in \mathcal{I}; \; h_j(\boldsymbol{x}) = 0 \; \forall \, j \in \mathcal{E} \tag{3}$$

Eq. (1) can be solved by general-purpose NO solvers such as `Knitro` [23], `Ipopt` [27], and `GENO` [17]. However, there are two caveats: (1) the above solvers only handle continuously differentiable objective and constraint functions, i.e., g, c_i's, and h_j's, but non-differentiable g, c_i's, and h_j's are common in Eq. (1), e.g., when d is the ℓ_1 or ℓ_∞ distance, or $f_{\boldsymbol{\theta}}$ uses non-differentiable

1. It should be "ascent" instead of "descent" due to the maximization, but we follow the `AutoAttack` package.
2. https://github.com/fra31/auto-attack

activations; (2) they require analytical gradients of g, c_i's, and h_j's, which are impractical to derive when DNN models $f_{\boldsymbol{\theta}}$ are involved.

`PyGRANSO`[3] [9, 18] is a recent `PyTorch`-port of the powerful MATLAB package `GRANSO` [9] which can handle general NO problems of form Eq. (3) and potentially with non-differentiable g, c_i's, and h_j's. It only requires these functions to be *almost everywhere differentiable*, which is satisfied by almost all forms of Eq. (1) proposed so far in the literature. `GRANSO` employs a quasi-Newton sequential quadratic programming (BFGS-SQP) to solve Eq. (3), and features a rigorous adaptive step-size rule via line search and a principled stopping criterion inspired by gradient sampling [3]. `PyGRANSO` equips `GRANSO` with auto-differentiation and GPU computing powered by `PyTorch`—crucial for deep learning problems. The stopping criterion is controlled by stationarity, total constraint violation, and optimization tolerance—all can be transparently controlled, but is typically unnecessary to tune. For the details of `PyGRANSO` package, please check: `https://arxiv.org/abs/2210.00973`.

3. `PyGRANSO` with constraint folding as a generic solver for Eq. (1)

Though `PyGRANSO` can serve as a promising solver for Eq. (1) with general metric d, we find in practice that naive deployment can suffer from slow convergence, or low quality solutions due to numerical issues. Below, we introduce `PyGRANSO` **W**ith **C**onstraint-**F**olding (PWCF), and other techniques that can substantially speed up the optimization process, and improve the solution quality.

3.1. Reformulating ℓ_∞ constraint to avoid sparse subgradients

The BFGS-SQP algorithm inside `PyGRANSO` relies on the subgradients of the objective and the constraint functions to approximate the (inverse) Hessian and to compute the search direction. Hence, when the subgradients are sparse, updating all optimization variables may take many iterations, leading to slow convergence. For the ℓ_∞ metric,

$$\partial_{\boldsymbol{z}} \|\boldsymbol{z}\|_\infty = \operatorname{conv}\{\boldsymbol{e}_k \operatorname{sign}(z_k) : z_k = \|\boldsymbol{z}\|_\infty \; \forall\, k\}, \tag{4}$$

where $\boldsymbol{e}_k$'s are the standard basis vectors, conv denotes convex hull, and $\operatorname{sign}(z_k) = z_k/|z_k|$ if $z_k \neq 0$, else $[-1, 1]$. The subgradient in Eq. (4) contains no more than $n_k = |\{k : z_k = \|\boldsymbol{z}\|_\infty\}|$ nonzeros, and hence is sparse when n_k is small. To avoid this issue, we propose a reformulation

$$\left\|\boldsymbol{x} - \boldsymbol{x}'\right\|_\infty \le \varepsilon \Longleftrightarrow -\varepsilon \mathbf{1} \le \boldsymbol{x} - \boldsymbol{x}' \le \varepsilon \mathbf{1}. \tag{5}$$

3.2. Constraint-folding to reduce the number of constraints

The natural image constraint $\boldsymbol{x}' \in [0, 1]^n$ is a set of n box constraints. The reformulation described in Section 3.1 introduces another $\Theta(n)$ box constraints. Although all these are just simple linear constraints, the $\Theta(n)$-growth is daunting: for natural images, n is the number of pixels that can easily get into hundreds of thousands. Typical NO problems become more difficult the number of constraints grows, e.g., leading to slow convergence for numerical algorithms.

To combat this, we introduce a folding technique that can reduce the number of constraints into a small constant. To see how this is possible, first note that any equality constraint $h(\boldsymbol{x}) = 0$ or

3. `https://ncvx.org`

inequality constraint $c(\boldsymbol{x}) \leq 0$ can be reformulated as

$$h(\boldsymbol{x}) = 0 \Longleftrightarrow |h(\boldsymbol{x})| \leq 0 \,, \quad c(\boldsymbol{x}) \leq 0 \Longleftrightarrow \max\{c(\boldsymbol{x}), 0\} \leq 0. \tag{6}$$

We can then fold them together as

$$\mathcal{F}(|h(\boldsymbol{x})|, \max\{c(\boldsymbol{x}), 0\}) \leq 0, \tag{7}$$

where $\mathcal{F} : \mathbb{R}_+^2 \mapsto \mathbb{R}_+$ ($\mathbb{R}_+ \doteq \{t : t \geq 0\}$) can be any function satisfying $\mathcal{F}(\boldsymbol{z}) = 0 \Longrightarrow \boldsymbol{z} = \mathbf{0}$, e.g., any ℓ_p ($p \geq 1$) norm.

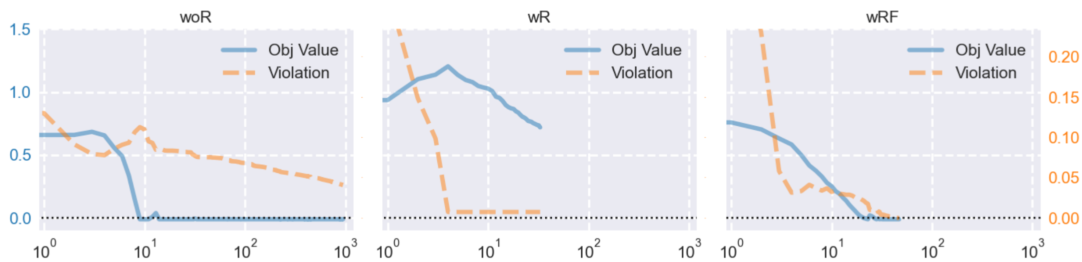

Figure 1: Optimization trajectory of the **objective value** and **constraint violation** w.r.t iterations for an ℓ_∞ case on CIFAR-10 dataset. **woR**: using ℓ_∞ original form; **wR**: with reformulation but no folding; **wRF**: with reformulation and folding. Maximum time budget per curve: $600s$ (only wRF terminates before reaching this budget). Both **objective** and **violation** reaching 0 indicates successful attack.

It is easy to verify the equivalence of Eq. (7) and the original constraints in Eq. (6). The folding technique can be used to a subset or all of the constraints; one can group and then fold constraints according to their physical meanings. We note that folding or aggregating constraints is not a new idea and has been popular in engineering design. For example, [21] uses ℓ_∞ folding and its log-sum-exponential approximation to deal with numerous design constraints. However, applying folding into NO problems in machine learning seems rare, potentially because producing non-differentiable constraint(s) due to the folding seems counterproductive.

In our experiments, we use $\mathcal{F} = \|\cdot\|_2$ to fold the $\Theta(n)$ box constraints from ℓ_∞ reformulation into a single constraint, enforce the $\boldsymbol{x}' \in [0,1]^n$ constraints in $f_{\boldsymbol{\theta}}$ by direct clipping. Fig. 1 shows clearly that combining folding and reformulation can substantially speed up convergence and boost the solution quality for our algorithm.

3.3. Loss clipping when solving Eq. (1) with PWCF

For Eq. (1) with the popular cross-entropy (CE) and margin losses, the objective value can easily dominate constraint violation during the maximization process. Since `PyGRANSO` tries to balance the objective value and constraint violation when making progress, it can persistently prioritize optimizing the objective over constraint satisfaction, resulting in very slow progress in finding a feasible solution. To resolve this numerical difficulty, we propose using clipped margin loss ℓ_{ML} with maximal value 0.01, as any $\ell_{ML} \geq 0$ indicates a successful attack. For the same reason, we use clipped CE loss with maximal value at 10 in PWCF[4].

4. Attack success happens when the true logit output less than $1/K$ (assuming softmax normalization is applied), where K is the number of classes. So the critical value is $-\log 1/K$, which is < 10 for $K \leq e^{10}$, sufficient for typical RE datasets.

4. Experiments and results: solving Eq. (1) with PWCF

4.1. PWCF offers competitive and complementary attack performance to Eq. (1)

We take SOTA ℓ_1-, ℓ_2-, and ℓ_∞-adversarially trained models on CIFAR10[5][6], and an adversarially-trained model with respect to the LPIPS distance[7] on ImageNet [16][8], to compare the attack performance by solving Eq. (1) between PWCF and the APGD[9] [6] method from `AutoAttack` package. The attack radii ε's are set following the common practice of adversarial RE[10].

Table 1: **Comparison of our PWCF with SOTA attack methods on ℓ_1-, ℓ_2- and ℓ_∞- attacks.** For given pretrained models, we report the models' **clean** and **robust** accuracy—lower **robust** accuracy means more effective attacks. We test on both CE and margin loss for APGD and PWCF. Numbers are in $(\%)$. **Model - Attack** denotes the selection of the models and the type of the performed adversarial attacks and its ε.

			APGD		PWCF(ours)		Square	APGD
Dataset	**Model - Attack**	**Clean**	**CE**	**M**	**CE**	**M**	**M**	**+PWCF**
CIFAR10	P_1 [20] - $\ell_1(12)$	73.3	0.96	0.00	28.6	0.00	2.28	**0.00**
	$WRN_{\text{-70-16}}$ [12] - $\ell_2(0.5)$	94.7	81.8	81.1	81.8	81.0	87.9	**80.8**
	$WRN_{\text{-70-16}}$ [12] - $\ell_\infty(0.03)$	90.8	69.4	68.0	73.6	72.8	71.6	**67.1**
ImageNet100	$PAT_{\text{-Alex}}$ [16] - $\ell_2(4.7)$	75.0	42.7	44.0	42.8	44.5	63.1	**40.9**
	$PAT_{\text{-Alex}}$ [16] - $\ell_\infty(0.016)$	75.0	48.0	48.2	56.6	48.8	59.9	**45.2**

From Table 1, we can conclude that: (1) PWCF performs strongly and comparably to APGD on ℓ_1, ℓ_2 and ℓ_∞ attacks, especially using *margin loss* as the objective; (2) PWCF is weak on ℓ_1 and ℓ_∞ attacks using CE loss, likely due to the bad numerical scaling of the CE loss; (3) Combining all successful attack samples found by APGD and PWCF (APGD+PWCF) can further reduce the robust accuracy compared to any single APGD or PWCF attack—PWCF and APGD are complementary. Note that [4] also remarks that the diversity of solutions matters much more than the superiority of individual solvers, which is the reason why `AutoAttack` includes Square Attack–a zero-th order black-box attack method that does not perform strongly itself as shown in Table 1.

5. https://github.com/locuslab/robust_union/tree/master/CIFAR10
6. https://github.com/deepmind/deepmind-research/tree/master/adversarial_robustness
7. See Section 4.2 for details.
8. https://github.com/cassidylaidlaw/perceptual-advex
9. We implement the margin loss on top of `AutoAttack`.
10. E.g., https://robustbench.github.io/ for Cifar10 ℓ_2 and ℓ_∞; https://github.com/locuslab/robust_union for Cifar10 ℓ_1; [16] for ImageNet ℓ_2 and ℓ_∞.

Table 2: **Attack performance of PWCF with margin loss on general ℓ_p and non-ℓ_p metrics**. We report attack success rates (numbers are in %). We test on $\ell_{1.5}$, ℓ_8, and PAT; numbers on ℓ_1, ℓ_2, and ℓ_∞ are included for reference. Numbers below each rate in parenthesis are the perturbation radii.

	Special ℓ_p			**General ℓ_p**		
Model	ℓ_1	ℓ_2	ℓ_∞	$\ell_{1.5}$	ℓ_8	PAT
Clean	100	100	100	100	100	100
	(2400)	(6.09)	(0.01569)	(44.40)	(0.07)	(0.5)
PAT	49.7	40.7	35.2	100	100	100
	(2400)	(4.7)	(0.017)	(443.98)	(0.70)	(0.5)

4.2. PWCF works for general (almost everywhere) differentiable ℓ_p and non-ℓ_p distances

As highlighted in Section 2, a major limitation of the PGD based solvers is that they cannot handle distances other than ℓ_1, ℓ_2, and ℓ_∞[11]. By contrast, PWCF stands out as a convenient choice for general distances. To show this, we apply PWCF to solve Eq. (1) with $\ell_{1.5}$ and ℓ_8 distances. In addition, we also solve Eq. (1) with the LPIPS perceptual metric [16, 30], i.e., perceptual attack (PAT) with

$$d(\boldsymbol{x}, \boldsymbol{x}') \doteq ||\phi(\boldsymbol{x}) - \phi(\boldsymbol{x}')||_2 \,, \qquad \phi(\boldsymbol{x}) \doteq [\, \widehat{g}_1(\boldsymbol{x}), \ldots, \widehat{g}_L(\boldsymbol{x})\,] \tag{8}$$

where $\widehat{g}_1(\boldsymbol{x}), \ldots, \widehat{g}_L(\boldsymbol{x})$ are the vectorized intermediate feature maps from pretrained DNNs.

PWCF handles them seamlessly, as shown in Table 2. Here we do not strive to set the most reasonable perturbation radii, especially for $\ell_{1.5}$ and ℓ_8 that have not been tested before, and hence we also do not stress the attack rates. Our point is that *PWCF is able to handle these general ℓ_p distances*. Table 3 further summarizes the details of performing the perceptual attack with $\varepsilon = 0.5$. Existing methods to compare are Perceptual Projected Gradient Descent (PPGD), Lagrangian perceptual attack (LPA) and its variant fast Lagrangian perceptual attack (Fast-LPA) methods, all developed in [16], based on iterative linearization and projection (PPGD), or penalty method (LPA, Fast-LPA) respectively. In addition to the objective values and attack success rates, we also report their chances of finding infeasible solutions. As observed in Table 3, our PWCF is the clear winner.

5. Conclusion

In this paper, we propose PWCF to solve the maximization problem Eq. (1) in robustness evaluations, blending the SOTA constrained optimization solver `PyGRANSO` with constraint folding and other tweaks. Our experimental results show that 1) PWCF can provide competitive and complementary performance compared with the SOTA methods on ℓ_1, ℓ_2, and ℓ_∞ attacks; 2) PWCF can deal with general attack models such as ℓ_p with $p \geq 1$ and perceptual attacks, which are beyond the reach of existing PGD-based methods; 3) PWCF involves little to zero parameter-tuning and obtains reliable solutions based on a principled stopping criterion. Our preliminary experiments also show that the proposed PWCF is general enough to solve Eq. (2) with good quality, which we will present in forthcoming papers.

11. We do not consider ℓ_0 in this paper as it is not a norm, but we acknowledge that [5] targets at generating ℓ_0 attacks using PGD-based method.

Table 3: **Performance comparison of different methods solving PAT with the clipped CE and margin (M) loss**. **Viol.** reports the ratio of final solutions that violate constraints. **Succ.** is the ratio of all *feasible successful attacks* divided by *total number of samples*. The model we test is `pat_alexnet_0.5` [16]. Evaluation is performed on ImageNet-100 dataset.

	CE Objective		**Margin Objective**	
Method	**Viol.** (%) ↓	**Succ.** (%) ↑	**Viol.** (%) ↓	**Succ.** (%) ↑
Fast-LPA	73.8	3.54	41.6	56.8
LPA	**0.00**	80.5	**0.00**	97.0
PPGD	5.44	25.5	**0.00**	38.5
PWCF	0.62	93.6	**0.00**	100

References

[1] Dimitri Bertsekas. *Nonlinear Programming 3rd Edition*. Athena Scientific, 2016. ISBN 9781886529052.

[2] Anand Bhattad, Min Jin Chong, Kaizhao Liang, Bo Li, and David A Forsyth. Big but imperceptible adversarial perturbations via semantic manipulation. *arXiv preprint arXiv:1904.06347*, 1(3), 2019.

[3] James V Burke, Frank E Curtis, Adrian S Lewis, Michael L Overton, and Lucas EA Simões. Gradient sampling methods for nonsmooth optimization. *Numerical Nonsmooth Optimization*, pages 201–225, 2020.

[4] Nicholas Carlini, Anish Athalye, Nicolas Papernot, Wieland Brendel, Jonas Rauber, Dimitris Tsipras, Ian Goodfellow, Aleksander Madry, and Alexey Kurakin. On evaluating adversarial robustness. *arXiv:1902.06705*, February 2019.

[5] Francesco Croce and Matthias Hein. Sparse and imperceivable adversarial attacks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4724–4732, 2019.

[6] Francesco Croce and Matthias Hein. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. *ArXiv*, abs/2003.01690, 2020.

[7] Francesco Croce and Matthias Hein. Minimally distorted adversarial examples with a fast adaptive boundary attack. In *International Conference on Machine Learning*, pages 2196–2205. PMLR, 2020.

[8] Francesco Croce and Matthias Hein. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *International conference on machine learning*, pages 2206–2216. PMLR, 2020.

[9] Frank E Curtis, Tim Mitchell, and Michael L Overton. A bfgs-sqp method for nonsmooth, nonconvex, constrained optimization and its evaluation using relative minimization profiles. *Optimization Methods and Software*, 32(1):148–181, 2017.

[10] Logan Engstrom, Brandon Tran, Dimitris Tsipras, Ludwig Schmidt, and Aleksander Madry. Exploring the landscape of spatial robustness. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 1802–1811. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/engstrom19a.html.

[11] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*, 2015. URL http://arxiv.org/abs/1412.6572.

[12] Sven Gowal, Chongli Qin, Jonathan Uesato, Timothy Mann, and Pushmeet Kohli. Uncovering the limits of adversarial training against norm-bounded adversarial examples. *arXiv preprint arXiv:2010.03593*, 2020.

[13] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. In *International Conference on Learning Representations*, 2018.

[14] Hossein Hosseini and Radha Poovendran. Semantic adversarial examples. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 1614–1619, 2018.

[15] Cassidy Laidlaw and Soheil Feizi. Functional adversarial attacks. *Advances in neural information processing systems*, 32, 2019.

[16] Cassidy Laidlaw, Sahil Singla, and Soheil Feizi. Perceptual adversarial robustness: Defense against unseen threat models. In *ICLR*, 2021.

[17] Sören Laue, Matthias Mitterreiter, and Joachim Giesen. Geno–generic optimization for classical machine learning. *Advances in Neural Information Processing Systems*, 32, 2019.

[18] Buyun Liang, Tim Mitchell, and Ju Sun. NCVX: A general-purpose optimization solver for constrained machine and deep learning. *arXiv:2210.00973*, 2022.

[19] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[20] Pratyush Maini, Eric Wong, and Zico Kolter. Adversarial robustness against the union of multiple perturbation models. In *International Conference on Machine Learning*, pages 6640–6650. PMLR, 2020.

[21] JRRA Martins and Nicholas MK Poon. On structural optimization using constraint aggregation. In *VI World Congress on Structural and Multidisciplinary Optimization WCSMO6, Rio de Janeiro, Brasil*. Citeseer, 2005.

[22] Marius Mosbach, Maksym Andriushchenko, Thomas Trost, Matthias Hein, and Dietrich Klakow. Logit pairing methods can fool gradient-based attacks. *arXiv preprint arXiv:1810.12042*, 2018.

[23] Gianni Pillo and Massimo Roma. *Large-scale nonlinear optimization*, volume 83. Springer Science & Business Media, 2006.

[24] Maura Pintor, Fabio Roli, Wieland Brendel, and Battista Biggio. Fast minimum-norm adversarial attacks through adaptive norm constraints. *Advances in Neural Information Processing Systems*, 34, 2021.

[25] Gagandeep Singh, Timon Gehr, Matthew Mirman, Markus Püschel, and Martin Vechev. Fast and effective robustness certification. *Advances in neural information processing systems*, 31, 2018.

[26] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[27] Andreas Wächter and Lorenz T Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical programming*, 106 (1):25–57, 2006.

[28] Eric Wong, Frank R. Schmidt, and J. Zico Kolter. Wasserstein adversarial examples via projected sinkhorn iterations. *arXiv:1902.07906*, February 2019.

[29] Chaowei Xiao, Jun-Yan Zhu, Bo Li, Warren He, Mingyan Liu, and Dawn Song. Spatially transformed adversarial examples. In *International Conference on Learning Representations*, 2018.

[30] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018.

[31] Yihua Zhang, Guanhua Zhang, Prashant Khanduri, Mingyi Hong, Shiyu Chang, and Sijia Liu. Revisiting and advancing fast adversarial training through the lens of bi-level optimization. *arXiv:2112.12376*, December 2021.