Augmented Lagrangian Adversarial Attacks

Jérôme Rony* Eric Granger Marco Pedersoli Ismail Ben Ayed
ÉTS Montreal, Canada

Code: https://github.com/jeromerony/augmented_lagrangian_adversarial_attacks

Abstract

Adversarial attack algorithms are dominated by penalty methods, which are slow in practice, or more efficient distance-customized methods, which are heavily tailored to the properties of the distance considered. We propose a white-box attack algorithm to generate minimally perturbed adversarial examples based on Augmented Lagrangian principles. We bring several algorithmic modifications, which have a crucial effect on performance. Our attack enjoys the generality of penalty methods and the computational efficiency of distance-customized algorithms, and can be readily used for a wide set of distances. We compare our attack to state-of-the-art methods on three datasets and several models, and consistently obtain competitive performances with similar or lower computational complexity.

1. Introduction

The last few years have seen an arms race in adversarial attacks, where several methods have been proposed to find minimally perturbed adversarial examples. In most cases, adversarial example generation is stated as a constrained optimization, which seeks the smallest additive perturbation, according to some distance, to misclassify an input. Existing methods fall within two categories: attacks using *penalty* methods for constrained optimization, and distance-customized attacks leveraging the properties of the distance considered (typically ℓ_p-norms).

Penalties are a natural choice, as they transform a constrained-optimization problem into an unconstrained one. Within this category, the most notorious attack is the one from Carlini and Wagner [7] known for its ℓ_2 variant. Building on the work of Carlini and Wagner, several other attacks have been proposed, *e.g.* EAD for ℓ_1 [9] and StrAttack [31], which generalizes EAD and introduces sparsity in the perturbations. More recently, other works have tackled other distances than the standard ℓ_p-norms, such as SSIM [16], CIEDE2000 [37] or LPIPS [21], and followed similar penalty-based strategies. Although convenient and applicable to a wide class of distances, penalty methods are known to result in slow convergence in the general field of optimization [18]. Furthermore, choosing the weight of the penalty is not a trivial task [19]. In fact, in adversarial attacks, it has recently been shown that the optimal penalty weight actually varies by orders of magnitude across samples and models [26]. Although penalty methods can achieve competitive performance, they typically require an expensive line-search to find optimal penalty weights [7], thereby requiring large numbers of iterations. This may impede their practical deployment for training robust models and efficiently evaluating robustness.

To accelerate the generation of adversarial examples, and improve performance over penalty methods, there has been an intensive focus on developing efficient algorithms customized for specific ℓ_p-norms [5, 12, 22, 25, 26]. However, such distance-customized methods are not generally applicable because they rely heavily on the geometry/properties of the distance considered (*e.g.* using projections and dual norms) to find minimally perturbed adversarial examples. The most notable attacks within this second category are: DeepFool for the ℓ_2 and ℓ_∞ norms, which uses a linear approximation of the model at each iteration [22]; DDN for the ℓ_2-norm, which uses projections on the ℓ_2-ball to decouple the direction and the norm of the perturbation [26]; and FAB for the ℓ_1, ℓ_2 and ℓ_∞, which combines a linear approximation of the model and projections w.r.t. the norm considered [12]. The recently proposed FMN attack [25] extends the DDN attack to other norms. Beyond ℓ_p-norms, Wong *et al.* proposed an attack [30] to produce adversarial perturbations with minimal Wasserstein distance using projected Sinkhorn iterations. Finally, one attack that does not strictly fall in either of the two categories was proposed by Brendel *et al.*, and designed for ℓ_p-norms with $p \in \{0, 1, 2, \infty\}$ [5]. In this attack, the optimization is formulated such that the perturbation follows the decision boundary of a classifier, while minimizing the considered distance. This is not limited to ℓ_p-norms but, in practice, the implementation leverages a trust-region solver designed for each ℓ_p-norm specifically, which limits applicability to other distances.

Penalty methods are generally applicable, and can be used for distances other than the standard ℓ_p-norms; for instance, CIEDE2000 [37] or LPIPS [21, 35]. They replace constrained problems with unconstrained ones by adding a penalty, which increases when the constraint is violated. A weight of the penalty is chosen and increased heuristically,

*Corresponding author: jerome.rony.1@etsmtl.net

while the unconstrained optimization is repeated several times. Powerful Augmented Lagrangian principles have well-established advantages over penalties in the general context of optimization [2, 3, 15, 24], and completely avoid penalty-weight heuristics by automatically estimating the multipliers. Furthermore, the multiplier estimates tend to the Lagrange multipliers, which avoids the ill-conditioning problems often encountered in penalty methods [2, 10]. Finally, Augmented Lagrangian methods avoid the need for explicitly solving the dual problem, unlike basic Lagrangian-dual optimization, which might be intractable/unstable for non-convex problems.

Despite their well-established advantages and popularity in the optimization community for solving non-convex problems, Augmented Lagrangian methods have not been investigated previously for adversarial attacks. While this seems rather surprising, we found in this work that the vanilla Augmented Lagrangian methods are not competitive in the context of adversarial attacks (*e.g.* in terms of computational efficiency), which might explain why they have been avoided so far. However, we introduce several algorithmic modifications to design a customized Augmented Lagrangian algorithm for adversarial example generation. Our modifications are crucial to achieve highly competitive performances in comparisons to state-of-the-art methods. The modifications include integrating the Augmented Lagrangian inner and outer iterations with joint updates of the perturbation and the multipliers, relaxing the need for an inner-convergence criterion, introducing an exponential moving average of the multipliers, and adapting the learning rates to the distance function. All-in-all, we propose a white-box Augmented Lagrangian Method for Adversarial (ALMA) attacks, which enjoys both the general applicability of penalty approaches and computational efficiency of distance-customized methods. Our attack can be readily used to generate adversarial examples for a large set of distances, including ℓ_1-norm, ℓ_2-norm, CIEDE2000, LPIPS and SSIM, and we advocate its use for other distances that might be investigated in future research in adversarial attacks. We evaluate our attack on three datasets (MNIST, CIFAR10 and ImageNet) and several models (regularly and adversarially-trained). For each distance, we compare our method against state-of-the-art attacks proposed specifically for that distance, and consistently observe competitive performance.

2. Preliminaries

Let $\boldsymbol{x}$ be a sample from the input space $\mathcal{X} \subset \mathbb{R}^d$, and $y \in \mathcal{Y}$ its associated label, where $\mathcal{Y}$ is a set of discrete labels of size K. Let $f : \mathbb{R}^d \to \mathbb{R}^K$ be a model that outputs logits (*i.e.* pre-softmax scores) $\boldsymbol{z} \in \mathbb{R}^K$ given an input $\boldsymbol{x}$; $f_k(\boldsymbol{x})$ denotes the k-th component of the vector $f(\boldsymbol{x})$. In a classification scenario, the probability $p_y = P(y|x)$ is obtained using the $\mathrm{softmax}$ function: $p_y = \mathrm{softmax}_y(\boldsymbol{z})$. In this work, we assume that $\mathcal{X}$ is the hypercube $\mathcal{X} = [0, 1]^d$, which is general enough for computer vision applications.

2.1. Problem formulation

The problem of adversarial example generation has been mainly formulated in two ways. One way is to find adversarial examples satisfying a distance constraint:

$$\begin{aligned} \text{find} \quad \delta \quad \text{s.t.} \quad & \arg\max_k f_k(\boldsymbol{x}+\delta) \neq y \\ & D(\boldsymbol{x}+\delta, \boldsymbol{x}) \leq \epsilon;\ \boldsymbol{x}+\delta \in \mathcal{X} \end{aligned} \tag{1}$$

Alternatively, the objective is to find adversarial examples that are minimally distorted w.r.t a distance function D:

$$\begin{aligned} \min_\delta \quad D(\boldsymbol{x}+\delta, \boldsymbol{x}) \quad \text{s.t.} \quad & \arg\max_k f_k(\boldsymbol{x}+\delta) \neq y \\ & \boldsymbol{x}+\delta \in \mathcal{X} \end{aligned} \tag{2}$$

In this work, we are interested in solving Equation 2, which is equivalent to solving Equation 1 for every ϵ. Thus, it is a more general but more difficult problem.

2.2. Equivalent problem

In the general case, constraint $\boldsymbol{x}+\delta \in \mathcal{X}$ is not necessarily trivial. However, in our context, this corresponds to a box constraint: $\mathbf{0} \leq \boldsymbol{x}+\delta \leq \mathbf{1}$. We therefore handle it with a simple projection $\mathcal{P}_{[0,1]}$. For brevity, we will omit this constraint in the rest of the paper. In the above formulations, $\arg\max$ is not differentiable, and therefore not readily amenable to gradient-based optimization. We replace the $\arg\max$ constraint with an inequality constraint on the logits, as done in several works, most notably [7]:

$$\min_\delta \quad D(\boldsymbol{x}+\delta, \boldsymbol{x}) \quad \text{s.t.} \quad f_y(\boldsymbol{x}+\delta) - \max_{k \neq y} f_k(\boldsymbol{x}+\delta) < 0 \tag{3}$$

While more suited to gradients, this constraint is not scale invariant, as noted in [13]: extreme scaling of the logits may result in gradient masking. We use a slightly modified Difference of Logits Ratio (DLR) for this constraint [13] :

$$\mathrm{DLR}^+(\boldsymbol{z}, y) = \frac{\boldsymbol{z}_y - \max_{i \neq y} \boldsymbol{z}_i}{\boldsymbol{z}_{\pi_1} - \boldsymbol{z}_{\pi_3}} \tag{4}$$

where $\boldsymbol{z} = f(\boldsymbol{x})$ and π is the ordering of the elements of $\boldsymbol{z}$ in decreasing order. This loss is negative if and only if $\boldsymbol{x}$ is not classified as y, and its maximum is 1. Therefore, we solve the following optimization problem:

$$\min_\delta \quad D(\boldsymbol{x}+\delta, \boldsymbol{x}) \quad \text{s.t.} \quad \mathrm{DLR}^+(f(\boldsymbol{x}+\delta), y) < 0 \tag{5}$$

2.3. Distances

Most attacks in the literature measure the size of the perturbations in terms of ℓ_p-norms. In this work, we propose

an attack that can find minimally perturbed adversarial examples w.r.t. several distances. We limit this work to four common measures: the ℓ_1 and ℓ_2 norms, the CIEDE2000 [27] and the LPIPS distance [35]. The CIEDE2000 color difference [27] is a metric designed to assess the perceptual difference between two colors. It is widely used to evaluate color accuracy of displays and printed materials. This metric was designed to be aligned with the perception of the human eye. Generally, a value smaller than 1 means that the color difference is imperceptible, between 1 and 2 is perceptible through close examination, between 2 and 10 is perceptible at a glance, and above 10 means that the colors are different. This metric is calculated in the CIELAB color space, so a conversion is needed (see Appendix B). The CIEDE2000 is defined between two color pixels, so we use the image level accumulated version as in [37]. The LPIPS distance [35] is a recently proposed perceptual metric based on the distance between deep features of two images for a chosen model. Zhang *et al*. showed that this distance aligns with human perception better than other perceptual similarity metrics such as SSIM. We use the LPIPS with AlexNet as in [21].

3. Methodology

3.1. General Augmented Lagrangian algorithm

To describe a minimization problem with one inequality constraint, we use the following notation:

$$\text{minimize} \quad g(\boldsymbol{x}) \quad \text{subject to} \quad h(\boldsymbol{x}) < 0 \tag{6}$$

with $g : \mathbb{R}^d \to \mathbb{R}$ the *objective function* and $h : \mathbb{R}^d \to \mathbb{R}$ the *inequality-constraint function*.

Penalty methods trade a constrained problem (6) with an unconstrained one by adding a term (penalty), which increases when the constraint is violated. A weight of the penalty, λ, is chosen heuristically, and the unconstrained optimization is repeated several times with increasing values of λ, until the constraint is satisfied. Augmented Lagrangian methods have well-established advantages over penalty methods in the general context of optimization [2, 3, 15, 24], and avoid completely such heuristics. They estimate automatically the multipliers, which yields adaptive and optimal weights for the constraints. Such estimates tend to the Lagrange multiplier, which avoids the ill-conditioning problems often encountered in penalty methods [2, 10]. Augmented Lagrangian methods also avoid the need to explicitly solve the dual problem, unlike basic Lagrangian optimization, which may be intractable/unstable for non-convex problems. From a more practical standpoint, the efficiency of Augmented Lagrangian methods depends solely on the ability to solve the inner minimization (which we detail below), making the implementation simpler and more robust. Despite these advantages and their popularity in the optimization community, Augmented Lagrangian methods have not (to our knowledge) been investigated for adversarial attacks. In the following, we customize Augmented Lagrangian principles to solve adversarial-attack problems of the form (2).

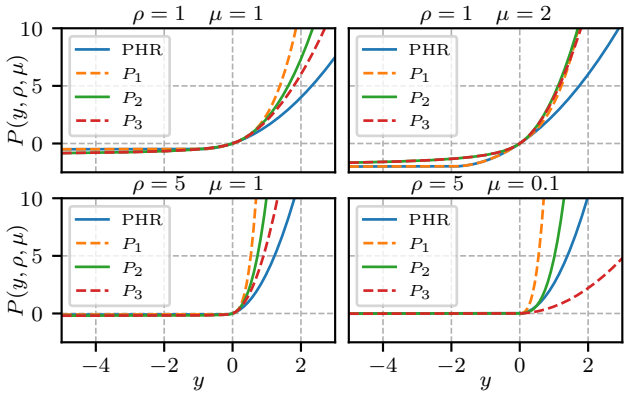

Figure 1: Examples of penalty-Lagrangian functions for different values of ρ and μ. The plotted functions are defined in [4] and given in Appendix D.

In general, an Augmented Lagrangian algorithm uses a succession of unconstrained optimization problems, each solved approximately. The algorithm can be broken down in two types of iterations: *outer* iterations indexed by i and *inner* iterations. During the inner iterations, the following Augmented Lagrangian function:

$$G(\boldsymbol{x}) = g(\boldsymbol{x}) + P(h(\boldsymbol{x}), \rho^{(i)}, \mu^{(i)}) \tag{7}$$

is approximately minimized w.r.t. to $\boldsymbol{x}$, using the previous solution as initialization, up to an inner convergence criterion. $P : \mathbb{R} \times \mathbb{R}_+^* \times \mathbb{R}_+^* \to \mathbb{R}$ is a *penalty-Lagrangian function* such that $P'(y, \rho, \mu) = \frac{\partial}{\partial y} P(y, \rho, \mu)$ exists and is continuous for all $y \in \mathbb{R}$ and $(\rho, \mu) \in (\mathbb{R}_+^*)^2$. Any candidate function P should satisfy these four axioms [4]:

Axiom 1. $\forall y \in \mathbb{R}, \forall (\rho, \mu) \in (\mathbb{R}_+^*)^2, \ \frac{\partial}{\partial y} P(y, \rho, \mu) \geq 0$

Axiom 2. $\forall (\rho, \mu) \in (\mathbb{R}_+^*)^2, \ \frac{\partial}{\partial y} P(0, \rho, \mu) = \mu$

Axiom 3. If, for all $j \in \mathbb{N}$, $0 < \mu_{\min} \leq \mu^{(j)} \leq \mu_{\max} < \infty$, then: $\lim_{j \to \infty} \rho^{(j)} = \infty$ and $\lim_{j \to \infty} y^{(j)} = y > 0$ imply that $\lim_{j \to \infty} \frac{\partial}{\partial y} P(y^{(j)}, \rho^{(j)}, \mu^{(j)}) = \infty$

Axiom 4. If, for all $j \in \mathbb{N}$, $0 < \mu_{\min} \leq \mu^{(j)} \leq \mu_{\max} < \infty$, then: $\lim_{j \to \infty} \rho^{(j)} = \infty$ and $\lim_{j \to \infty} y^{(j)} = y < 0$ imply that $\lim_{j \to \infty} \frac{\partial}{\partial y} P(y^{(j)}, \rho^{(j)}, \mu^{(j)}) = 0$

The first and second axioms guarantee that the derivative of the penalty-Lagrangian function P is positive and equal to μ when $y = 0$. The third and fourth axioms guarantee that the derivative of P w.r.t. y tends to infinity if the constraint is not satisfied (*i.e.* $h(\boldsymbol{x}) \geq 0$), and tends to 0 if the constraint is satisfied (*i.e.* $h(\boldsymbol{x}) < 0$). Figure 1 contains examples

of widely used penalty-Lagrangian functions. Once the inner convergence criterion is satisfied, an outer iteration is performed, during which the *penalty multiplier* μ and the *penalty parameter* ρ are modified. The penalty multiplier μ is updated to the derivative of P at the current value w.r.t. the constraint function:

$$\mu^{(i+1)} = P'(h(\boldsymbol{x}), \rho^{(i)}, \mu^{(i)}) \tag{8}$$

This means that the penalty multiplier increases when the constraint is not satisfied and, otherwise, is reduced. This can be seen as an *adaptive* way of choosing the penalty weight in a penalty method. The penalty parameter ρ is increased based on the value of the constraint function at the current outer iteration, compared to the previous one. If the constraint function has not improved (*i.e.* reduced) significantly, then ρ is multiplied by a fixed factor, typically between 2 and 100 [4]. With higher values of the penalty parameter ρ, the penalty-Lagrangian function tends to an ideal penalty, as can be seen in Figure 1. Algorithm 1 describes a generic Augmented Lagrangian method.

Algorithm 1 Generic Augmented Lagrangian method

Require: Function to minimize f, constraint function g
Require: Initial value $\boldsymbol{x}^{(0)}$
Require: Penalty function P, initial multiplier $\mu^{(0)}$, $\rho^{(0)}$
for $i \leftarrow 0$ to $N-1$ **do**
  Using $\boldsymbol{x}^{(i)}$ as initialization, minimize (approximately): $G(\boldsymbol{x}) = g(\boldsymbol{x}) + P(h(\boldsymbol{x}), \rho^{(i)}, \mu^{(i)})$
  $\boldsymbol{x}^{(i+1)} \leftarrow$ approximate minimizer of G
  $\mu^{(i+1)} \leftarrow P'(h(\boldsymbol{x}^{(i+1)}), \rho^{(i)}, \mu^{(i)})$
  if the constraint does not improve **then**
    Set $\rho^{(i+1)} > \rho^{(i)}$
  else
    $\rho^{(i+1)} \leftarrow \rho^{(i)}$
  end if
end for

3.2. Augmented Lagrangian Attack

We propose to use Augmented Lagrangian methods to solve (5). However, we found that a vanilla Augmented Lagrangian algorithm is not well-suited for adversarial attacks. Indeed, alternating between inner and outer iterations is rather slow compared to the few hundreds iterations in typical adversarial attacks. Moreover, we wish to obtain an algorithm with a fixed number of iterations for practical purposes. This is clearly incompatible with the use of an inner convergence criterion required to stop the approximate minimization of G (step 2 of Algorithm 1). Designing a good inner convergence criterion is not a trivial task either. Therefore, we propose a modification of the traditional Augmented Lagrangian algorithm. We combine the inner and outer iterations, resulting in a joint update of the perturbation $\delta = \tilde{\boldsymbol{x}} - \boldsymbol{x}$ and the penalty multiplier μ. This means that we do not need an inner convergence criterion as we are always adapting μ. This also requires to adapt the increase of ρ, which depends on the value of the inequality-constraint function. Algorithm 2 presents our approach and, in the following sections, we detail several important design choices for the algorithm.

Algorithm 2 ALMA attack

Require: Classifier f, original image $\boldsymbol{x}$, true or target label y
Require: Number of iterations N, initial step size $\eta^{(0)}$, penalty parameter increase rate $\gamma > 1$, constraint improvement rate $\tau \in [0, 1]$, M number of steps between ρ increase.
Require: D distance function
Require: Penalty function P, initial multiplier $\mu^{(0)}$, initial penalty parameter $\rho^{(0)}$
Initialize $\tilde{\boldsymbol{x}}^{(0)} \leftarrow \boldsymbol{x}$,
for $i \leftarrow 0$ to $N-1$ **do**
  $\boldsymbol{z} \leftarrow f(\tilde{\boldsymbol{x}}^{(i)})$
  $d^{(i)} \leftarrow \mathrm{DLR}^{+}(\boldsymbol{z}, y)$ ▷ tDLR^{+} for targeted attack
  $\hat{\mu} \leftarrow \nabla_d P(d^{(i)}, \rho^{(i)}, \mu^{(i)})$ ▷ New penalty multiplier
  $\mu^{(i+1)} \leftarrow \mathcal{P}_{[\mu_{\min}, \mu_{\max}]}[\alpha\mu^{(i)} + (1-\alpha)\hat{\mu}]$ ▷ EMA
  $L \leftarrow D(\tilde{\boldsymbol{x}}^{(i)}, \boldsymbol{x}) + P(d^{(i)}, \rho^{(i)}, \mu^{(i+1)})$ ▷ Loss
  $\boldsymbol{g} \leftarrow \nabla_{\tilde{\boldsymbol{x}}} L$ ▷ Gradient of loss w.r.t. $\tilde{\boldsymbol{x}}$
  $\tilde{\boldsymbol{x}}^{(i+1)} \leftarrow \mathcal{P}_{[0,1]}[\tilde{\boldsymbol{x}}^{(i)} - \eta^{(i)}\boldsymbol{g}]$ ▷ Step and box-constraint
  if $(i+1) \bmod M = 0$ and $d^{(j)} > 0, \forall j \in \{0, \dots, i\}$ **and** $d^{(i)} > \tau d^{(i-M)}$ **then**
    $\rho^{(i+1)} \leftarrow \gamma\rho^{(i)}$ ▷ If no adversarial has been found and d does not decrease significantly, increase ρ by a factor of γ
  else
    $\rho^{(i+1)} \leftarrow \rho^{(i)}$
  end if
end for
Return $\tilde{\boldsymbol{x}}^{(i)}$ that is adversarial and has smallest $D(\tilde{\boldsymbol{x}}^{(i)}, \boldsymbol{x})$

3.2.1 Penalty parameters adaptation

The most critical design choices for our attack are the adaptation of the penalty parameters μ and ρ.

Penalty multiplier. μ is usually modified in each outer iteration (after the inner problem is approximately solved), and taken as the derivative of the penalty function w.r.t. the constraint function (see Equation 8). In our algorithm, we modify μ at each iteration. This approach aims at reducing the budget needed for the optimization by combining inner and outer iterations. However, this leads to spiking values of μ, which tend to make the optimization unstable. Figure 2 shows the evolution of μ during the attack, as well as the ℓ_2-norm of the perturbation, when attacking (with ALMA ℓ_2) a single MNIST sample with the SmallCNN model. We can see that μ regularly spike to high values when not using EMA, which in turns increases the ℓ_2-norm of the perturbation because the penalty term dominates. To solve this issue, we smooth the values of μ using an Exponential Moving

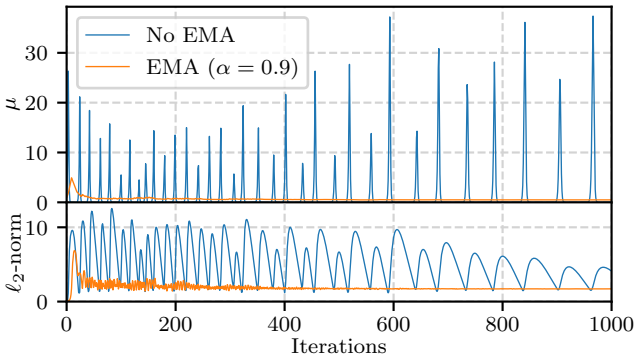

Figure 2: Example of the evolution of the penalty multiplier μ and the ℓ_2-norm of the perturbation during the optimization when attacking a single MNIST sample for the SmallCNN model. This leads to a ℓ_2-norm of the final adversarial perturbation of 2.13 without EMA and 1.71 with EMA ($\alpha = 0.9$). The norm of the perturbation can be lower without EMA in some iterations, but it is associated with an increase in μ, meaning that the perturbed sample $\tilde{\boldsymbol{x}}^{(i)}$ is not adversarial.

Average (EMA) of μ:

$$\begin{aligned} \hat{\mu} &= P'(g(\boldsymbol{x}), \rho^{(i)}, \mu^{(i)}) \\ \mu^{(i+1)} &= \alpha\mu^{(i)} + (1-\alpha)\hat{\mu} \end{aligned} \tag{9}$$

where $\alpha \in \mathbb{R}$ is a hyper-parameter. This corresponds to steps 5 and 6 in Algorithm 2. When using an EMA with $\alpha = 0.9$, the evolution of μ is smoother, with a much smaller spike at the beginning and then stabilizing. As a consequence, the ℓ_2-norm of the final perturbation is smaller when using an EMA at 1.71 for $\alpha = 0.9$, compared to 2.13 when $\alpha = 0$.

For numerical stability, we also need to constrain the value of the penalty multiplier such that it never becomes too small or too large. Therefore, we also project μ on the safeguarding interval $[\mu_{\text{min}}, \mu_{\text{max}}] \subset \mathbb{R}_+^*$.

Penalty parameter. ρ is typically increased in each outer iteration if the constraint has not improved during the inner minimization. Since inner and outer iterations are combined in our attack, we adopt a simple strategy to adapt ρ. Every M iterations, we verify if an adversarial example was found and, if not, if the constraint has improved. If no adversarial example was found, and the constraint does not improve (*e.g.* reduced by 5% in the last M iterations), we set $\rho^{(i+1)} = \gamma\rho^{(i)}$. This corresponds to steps 10 to 14 of Algorithm 2.

3.2.2 Choice of penalty function

Experiments have shown that the choice of the penalty function is of great importance, especially when considering non-convex problems [4]. Many functions have been proposed in the literature. In our experiment, we use the P_2 penalty function proposed in [20], which is defined as followed:

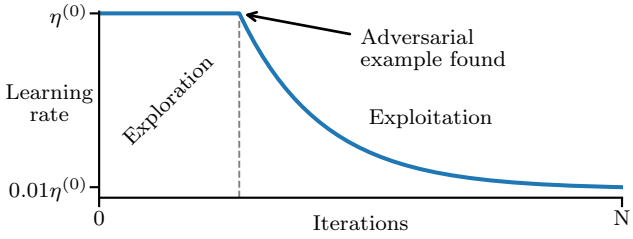

Figure 3: Exponential learning rate decay for the attack.

$$P_2(y, \rho, \mu) = \begin{cases} \mu y + \mu\rho y^2 + \frac{1}{6}\rho^2 y^3 & \text{if} \quad y \geq 0 \\ \frac{\mu y}{1-\rho y} & \text{if} \quad y < 0 \end{cases} \tag{10}$$

where y represents the value of the constraint: $\text{DLR}^+(\boldsymbol{z}, y)$ in our case (or $\text{tDLR}^+(\boldsymbol{z}, y)$ in a targeted scenario). In the numerical comparison of several penalty-Lagrangian functions done by Birgin *et al.* [4], P_2 is second to PHRQuad in terms of robustness. However, we found that P_2 is more suited to our problem. Experimentally, PHRQuad's derivative is smaller than P_2, resulting in smaller increase of μ in each iteration. This, in turn, results in an increase of ρ because no adversarial example is found (see step 10 and 11 of Algorithm 2). Generally, increasing ρ helps in finding a feasible point (*i.e.* adversarial examples), at the cost of a larger final distance for the perturbation. The choice of P_2 also depends on the algorithm design. In our attack, we chose to increase ρ every M steps if no adversarial example has been found, regardless of the usual convergence criterion used in more traditional Augmented Lagrangian methods. Therefore, we need a penalty function with an "aggressive" derivative such that μ is modified quickly.

3.2.3 Learning rate scheduling

Initial learning rate. Different distance functions can have widely different scales (see Tables 1 and 2, median results column). Therefore, the initial learning rate is chosen adaptively for each sample $\boldsymbol{x}$, such that the first gradient step (step 9 in Algorithm 2) increases D by ϵ, or formally, such that: $D(\tilde{\boldsymbol{x}}^{(1)}, \boldsymbol{x}) = \epsilon$. To obtain that value, we compute the gradient $\boldsymbol{g}$ of DLR^+ w.r.t. $\boldsymbol{x}$, and find the scalar $\eta^{(0)}$ such that $D(\mathcal{P}_{[0,1]}[\boldsymbol{x} - \eta^{(0)}\boldsymbol{g}], \boldsymbol{x}) = \epsilon$, using a line search followed by a binary search.

Learning rate decay. The optimization process can be partitioned in two phases: exploration and exploitation. The exploration phase corresponds to finding a feasible point (*i.e.* an adversarial example) for Equation 2. Then, the exploitation phase consists in refining the feasible point, *i.e.* finding a minimally distorted adversarial example. In our algorithm, the learning rate remains constant during the

exploration phase, and decays exponentially during the exploitation phase to reach a final learning rate of $\eta^{(0)}/100$. Figure 3 illustrates the scheduling of the learning rate decay.

3.2.4 Optimization algorithm

It is well known that gradient descent has a slow convergence rate, especially for ill-conditioned problem. Several optimization algorithms have been proposed to accelerate first-order methods, such as the momentum method or Nesterov's acceleration [23]. One such algorithm is RMSProp [28], in which the learning rate is divided by the square root of the EMA of the squared gradient. We use RMSProp combined with the momentum method in our attack, with a slight modification. Originally, the EMA of the squared gradient is initialized at 0. To avoid dividing by a small value in the first iteration (which would result in a large increase of the distance during the first iteration), we initialize the EMA of the squared gradient at 1. For simplicity, Algorithm 2 only describes a regular gradient descent update at step 9.

4. Experiments

Datasets. To evaluate our attack, we conduct experiments on three datasets: MNIST, CIFAR10 and ImageNet with two different budgets: 100 and 1000 iterations. On MNIST, we test the attacks on the whole test set. On CIFAR10 and ImageNet, we test the attacks on 1000 samples. These 1000 samples correspond to the first 1000 samples of the test set on CIFAR10 and 1000 randomly chosen samples from the validation set for ImageNet.

Models. It has been observed that some attacks can work well for regularly trained models, and fail against defended (*i.e.* adversarially trained) models [6, 13, 26]. Therefore, for each dataset, we evaluate the attack against a collection of models. Specifically, for MNIST, we use four models. The first three are the same model as in [7, 25, 26] with three training schemes: a regularly trained model we call SmallCNN, a ℓ_2-adversarially trained model from [26] we call SmallCNN-DDN and a ℓ_∞-adversarially trained model from [34] we call SmallCNN-TRADES. To vary architectures, we also test on the ℓ_∞-adversarially trained model from [33] we call CROWN-IBP. This is the large variant trained with $\epsilon = 0.4$. For CIFAR10, we use three models: a regularly trained Wide ResNet 28-10 [32], a ℓ_∞-adversarially trained Wide ResNet 28-10 from [8] and a ℓ_2-adversarially trained ResNet-50 [17] from [1]. These models are fetched from the RobustBench library [11]. On ImageNet, we test the attacks on three ResNet-50 models: one regularly trained, and two adversarially trained (ℓ_2 and ℓ_∞), available through the robustness library [14]. All the models have been selected because they have obtained high robust accuracies in third-party evaluations [13].

Metrics. For each attack, we report the Attack Success Rate (ASR) which is the proportion of examples for which an adversarial perturbation was found, and the median perturbation size. To compare the complexities of the attacks, we use the average number of forward and backward model propagations per sample needed to perform each attack. In the deep learning context, the number of model propagations (forward and backward) is a good proxy for the complexity of an attack, as these operations generally require orders of magnitude more computations than the other operations of an attack. We also prefer this measure to the attack run-time since this makes for comparisons that are independent of both software optimizations and hardware differences.

Hyperparameters. For our attack, we consider two budgets: 100 and 1000 iterations. The goal of this work is to propose a framework to generate minimally perturbed adversarial example w.r.t. any distance. Therefore, we chose one α (*i.e.* the coefficient of the EMA) per budget and one ϵ (*i.e.* the increase of D in the first iteration, see Section 3.2.3) per distance. α and ϵ are shared accross all datasets and models. For the 100 iterations budget, we set $\alpha = 0.5$ and for the 1000 iterations budget, we set $\alpha = 0.9$. For ϵ, we chose an initial value in relation to the usual scale of each distance: typically a tenth of the expected distances between the adversarial examples and the original inputs works well. Although it requires prior knowledge on the expected distances, a good ϵ can quickly be found when experimenting with a new distance function. Appendix F gives the initial values of ϵ used in our experiments for each distance. The other hyperparameters are fixed. We use $\mu^{(0)} = \rho^{(0)} = 1$, $\mu_{\text{min}} = 10^{-6}$ and $\mu_{\text{max}} = 10^{12}$, $\gamma = 1.2$, $\tau = 0.95$ and $M = 10$ in all our experiments. These values are not usual for Augmented Lagrangian algorithms; they reflect our design choice of combining inner and outer iterations. Since we update our penalty parameters (*i.e.* μ and ρ) more frequently, γ can have a smaller value.

Attacks. We compare our attack with various state-of-the-art attacks from the literature. Specifically, we evaluate the ℓ_1 variant of our attack against the EAD attack [9] with budgets of 9×100 and 9×1000, and FAB ℓ_1 [12] and FMN ℓ_1 [25] attacks with budgets of 100 and 1000 iterations. For the ℓ_2 variant of our attack, we compare it to the C&W ℓ_2 [7] attack with budgets of 9×1000 and $9\times10\,000$, DDN [26], FAB ℓ_2 [12] and FMN ℓ_2 [25] all with budgets of 100 and 1000 iterations.[1] For FAB ℓ_1 and ℓ_2 on ImageNet, we use the targeted variant of the attack, as was done in the original work.[2] We compare the CIEDE2000 variant of our

[1] We tried to include the B&B ℓ_1 and ℓ_2 attacks [5] in our comparison, but their official implementations kept crashing. In the cases where it did not, we observed median distances worse than FAB. As such, we omitted the method completely from the experiments.

[2] The FAB attack needs to compute the Jacobian of the output of the model w.r.t. to the input. This increases the complexity by a factor K (*i.e.* the number of classes), making it impractical for datasets with large number of classes, such as ImageNet. See Section 4 of [12].

Distance	Attack	ASR (%)	Median Distance	Forwards / Backwards
ℓ_1-norm	EAD 9×100 [9]	97.18	21.94	807 / 407
	EAD 9×1000 [9]	97.76	19.19	5 025 / 2 516
	FAB ℓ_1 100 [12]	99.80	27.41	201 / 1 000
	FAB ℓ_1 1000 [12]	99.83	24.21	2 001 / 10 000
	FMN ℓ_1 100 [25]	69.87	–	100 / 100
	FMN ℓ_1 1000 [25]	95.35	7.34	1 000 / 1 000
	ALMA ℓ_1 100	99.90	11.45	100 / 100
	ALMA ℓ_1 1000	100	7.16	1 000 / 1 000
ℓ_2-norm	C&W ℓ_2 9×1000 [7]	40.19	–	8 643 / 8 643
	C&W ℓ_2 9×10 000 [7]	40.20	–	85 907 / 85 907
	DDN 100 [26]	98.48	1.86	100 / 100
	DDN 1000 [26]	99.83	1.61	1 000 / 1 000
	FAB ℓ_2 100 [12]	83.25	2.10	201 / 1 000
	FAB ℓ_2 1000 [12]	96.28	1.77	2 001 / 10 000
	FMN ℓ_2 100 [25]	70.99	3.18	100 / 100
	FMN ℓ_2 1000 [25]	96.02	1.77	1 000 / 1 000
	$\text{APGD}^{\text{T}}_{\text{DLR}}$ ℓ_2‡ [13]	99.98	2.52	12 271 / 12 253
	ALMA ℓ_2 100	99.72	2.38	100 / 100
	ALMA ℓ_2 1000	100	1.61	1 000 / 1 000

Table 1: Performance for attacks on the MNIST dataset. Geometric mean over the four models. For C&W ℓ_2, the attack fails on the IBP model (3% ASR), so the median distance is undefined. ‡A binary search is performed on each sample to get a minimal perturbation attack (Equation 2).

attack with, to the best of our knowledge, the only attack using this metric: the PerC-AL attack [37] with budgets of 100 and 1000 iterations. Lastly, for the LPIPS variant, we compare our attack to the LPA attack [21], which is designed to solve Equation 1 (*i.e.* find a perturbation within a distance budget). We perform a binary search on the distance budget to find the minimum budget for which an adversarial example can be found. We also add the $\text{APGD}^{\text{T}}_{\text{DLR}}$ ℓ_2 [13] to the ℓ_2 comparison with a binary search. For the binary seaarch, we start with a large enough budget (depending on the dataset) such that the attacks can reach 100% ASR and perform enough binary search steps to reach a precision of 0.01 for the ℓ_2-norm and 0.001 for the LPIPS.

Finally, we perform a C&W type attack as a baseline for the CIEDE2000 and LPIPS distances. We replace the ℓ_2-norm with the corresponding target distance and use a budget of 9×1000 which is enough to get a good performance while keeping an acceptable budget (compared to 9×10 000).

5. Results

To summarize the results, we report the geometric mean of each metric over the models considered for each dataset, in Table 1 for MNIST and Table 2 for CIFAR10 and ImageNet. Tables containing the detailed results for each model can be found in Appendix H. We also present robust accuracy curves, which reflect directly the performance in terms of minimum distance of the adversarial examples, by showing an expected robust accuracy as a function of a perturbation budget. For all curves, the dotted lines denote the

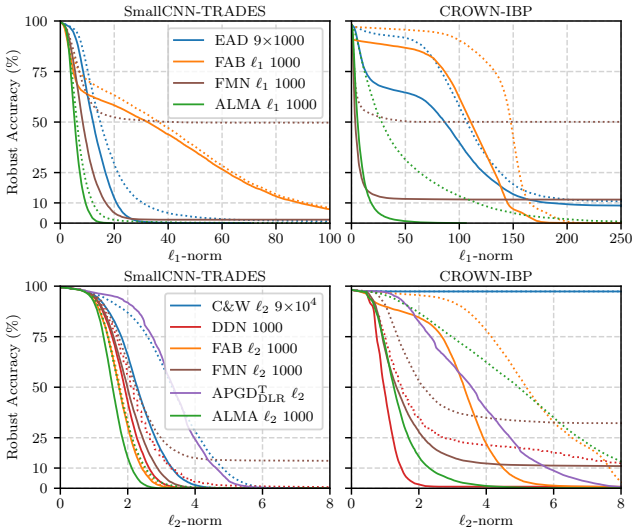

Figure 4: Robust accuracy curves for the SmallCNN-TRADES and CROWN-IBP adversarially trained models on MNIST against ℓ_1 (top row) and ℓ_2 (bottom row) attacks.

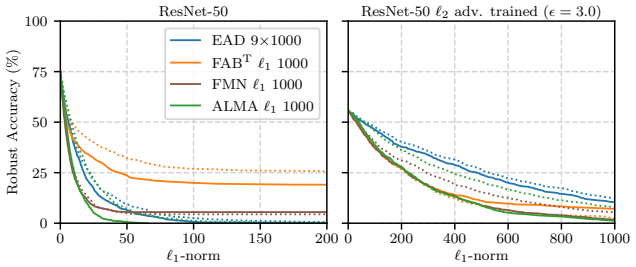

Figure 5: Robust accuracy curves for regular and ℓ_2-adversarial ResNet-50 on ImageNet against ℓ_1 attacks.

reduced budget attacks of the corresponding colors. With these curves, we can also find the median distance (see Appendix H) for each attack by looking at the distance for which the accuracy is 50%. All the robust accuracy curves can be found in Appendix I. Here we display a few for which there are significant differences between the attacks. For MNIST, we show the curves in Figure 4 for the SmallCNN-TRADES and CROWN-IBP models against ℓ_1 and ℓ_2 attacks. For ImageNet, we show the curves for ResNet-50 regularly and ℓ_2-adversarially trained from [14] against ℓ_1 attacks.

Across all datasets and models, our attack consistently obtains results competitive with attacks tailored to each distance. On MNIST, ALMA ℓ_1 is the only attack to reach an ASR of 100% with only FAB ℓ_1 outperforming it on the SmallCNN model. FMN ℓ_1 reaches a lower median ℓ_1 at the cost of a reduced ASR on the CROWN IBP model. ALMA ℓ_2 is only marginally outperformed by $\text{APGD}^{\text{T}}_{\text{DLR}}$ ℓ_2 on the SmallCNN and SmallCNN-DDN models, but again, ALMA is the only attack to consistently reach 100% ASR.

Distance	Attack	CIFAR10			ImageNet		
		ASR (%)	Median Distance	Forwards / Backwards	ASR (%)	Median Distance	Forwards / Backwards
ℓ_1-norm	EAD 9×100 [9] (AAAI'17)	100	6.11	572 / 290	100	13.87	488 / 248
	EAD 9×1000 [9] (AAAI'17)	100	5.44	4 284 / 2 146	100	12.83	3 758 / 1 883
	FAB† ℓ_1 100 [12] (ICML'20)	96.58	4.26	201 / 1 000	88.82	10.72	1 810 / 900
	FAB† ℓ_1 1000 [12] (ICML'20)	99.00	3.78	2 001 / 10 000	89.07	8.88	18 010 / 9 000
	FMN ℓ_1 100 [25]	99.90	3.64	100 / 100	94.33	8.43	100 / 100
	FMN ℓ_1 1000 [25]	99.83	3.54	1 000 / 1 000	93.93	7.58	1 000 / 1 000
	ALMA ℓ_1 100	100	4.31	100 / 100	100	19.79	100 / 100
	ALMA ℓ_1 1000	100	3.69	1 000 / 1 000	100	12.10	1 000 / 1 000
ℓ_2-norm	C&W ℓ_2 9×1000 [7] (SP'17)	100	0.40	7 976 / 7 974	99.83	0.57	7 248 / 7 246
	C&W ℓ_2 9×10 000 [7] (SP'17)	100	0.40	78 081 / 78 079	99.83	0.57	67 479 / 67 476
	DDN 100 [26] (CVPR'19)	100	0.43	100 / 100	99.70	0.51	100 / 100
	DDN 1000 [26] (CVPR'19)	100	0.42	1 000 / 1 000	99.87	0.50	1 000 / 1 000
	FAB† ℓ_2 100 [12] (ICML'20)	100	0.41	201 / 1 000	99.70	0.35	1 810 / 900
	FAB† ℓ_2 1000 [12] (ICML'20)	100	0.41	2 001 / 10 000	98.90	0.35	18 010 / 9 000
	FMN ℓ_2 100 [25]	99.90	0.43	100 / 100	99.43	0.38	100 / 100
	FMN ℓ_2 1000 [25]	99.83	0.40	1 000 / 1 000	99.63	0.36	1 000 / 1 000
	$\text{APGD}^{\text{T}}_{\text{DLR}}$ ℓ_2‡ [13] (ICML'20)	100	0.38	5 345 / 5 321	100	0.34	6 096 / 6 068
	ALMA ℓ_2 100	100	0.40	100 / 100	100	0.38	100 / 100
	ALMA ℓ_2 1000	100	0.38	1 000 / 1 000	100	0.35	1 000 / 1 000
CIEDE 2000	C&W CIEDE2000 9×1000	100	0.93	6 729 / 6 726	100	1.39	5 635 / 5 632
	PerC-AL 100 [37] (CVPR'20)	100	2.87	201 / 100	99.90	3.55	201 / 100
	PerC-AL 1000 [37] (CVPR'20)	100	2.72	2 001 / 1 000	99.93	3.42	2 001 / 1 000
	ALMA CIEDE2000 100	100	1.09	100 / 100	100	0.75	100 / 100
	ALMA CIEDE2000 1000	100	0.78	1 000 / 1 000	100	0.63	1 000 / 1 000
LPIPS $\times 10^{-2}$	C&W LPIPS 9×1000	100	0.47	6 658 / 6 655	100	2.07	4 950 / 4 944
	LPA‡ [21] (ICLR'21)	100	5.39	1 118 / 1 108	100	5.79	1 211 / 1 201
	ALMA LPIPS 100	99.97	2.47	100 / 100	100	1.59	100 / 100
	ALMA LPIPS 1000	100	0.60	1 000 / 1 000	100	1.13	1000 / 1000

Table 2: Performance for attacks on the CIFAR10 and ImageNet datasets. Geometric mean over the three models for each dataset. †For ImageNet, we use the targeted variant of the attack as in [12] (see Section 4). ‡A binary search is performed on each sample to get a minimal perturbation attack (Equation 2).

The results on MNIST get confirmed by the experiments on CIFAR10 and ImageNet. All the variants of ALMA consistently obtain 100% success rate (except for ALMA LPIPS 100). ALMA ℓ_1 obtains worse median norms than FAB ℓ_1 and FMN ℓ_1, but again, at the cost of a reduced ASR for both attacks. Surprisingly, FMN ℓ_1 obtains lower ASR for the higher budget variant, but with a lower median. For the ℓ_2 attacks, several perform similarly, with only $\text{APGD}^{\text{T}}_{\text{DLR}}$ ℓ_2 reaching a lower ℓ_2 median with a 100% ASR. This is expected given that this is a distance budget attack, with a much higher overall complexity when combining it with a binary search. For the CIEDE2000 distance, the PerC-AL attack [37] obtains significantly larger median distance on both CIFAR10 and ImageNet. Investigating the original code, we found errors in the implementation of the CIEDE2000, resulting in both wrong values and wrong gradients.[3]. The LPIPS variant of ALMA performs much better than LPA combined with a binary search for both datasets. Being a penalty base approach, our C&W baseline gets performance that is on par or better than ALMA LPIPS on both datasets, but at a much higher computational cost. However, it does not perform as well for the CIEDE2000 distance.

Overall, our attack offers a reliable trade-off between speed and performance in terms of ASR and perturbation size, given that the hyper-parameters are not tuned beyond ϵ, which is distance specific, and α which is directly related to the number of steps.

6. Conclusion

In this paper, an adversarial attack based on Augmented Lagrangian methods is proposed, which acts as a general framework for generating minimally perturbed adversarial examples w.r.t. several distances. In most of our experiments, it offers a good trade-off between performance and computational complexity in comparison to state-of-the-art methods. We believe that our general method could serve as a starting point for designing efficient attacks minimizing new distances.

[3]To verify this, we tested both implementations of the CIEDE2000 against the test values provided for this purpose in [27].

References

[1] Maximilian Augustin, Alexander Meinke, and Matthias Hein. Adversarial robustness on in-and out-distribution improves explainability. In *European Conference on Computer Vision*, 2020. 6, 14, 15, 17, 18, 19

[2] Dimitri P Bertsekas. *Constrained Optimization and Lagrange Multiplier Methods*. Academic press, 2014. 2, 3

[3] Dimitri P Bertsekas. *Nonlinear Programming: 3rd Edition*. Athena Scientific, 2016. 2, 3

[4] Ernesto G Birgin, Romulo A Castillo, and José Mario Martínez. Numerical comparison of augmented lagrangian algorithms for nonconvex problems. *Computational Optimization and Applications*, 31(1), 2005. 3, 4, 5, 12

[5] Wieland Brendel, Jonas Rauber, Matthias Kümmerer, Ivan Ustyuzhaninov, and Matthias Bethge. Accurate, reliable and fast robustness evaluation. In *Advances in Neural Information Processing Systems*, 2019. 1, 6

[6] Nicholas Carlini, Anish Athalye, Nicolas Papernot, Wieland Brendel, Jonas Rauber, Dimitris Tsipras, Ian Goodfellow, Aleksander Madry, and Alexey Kurakin. On evaluating adversarial robustness. *arXiv preprint arXiv:1902.06705*, 2019. 6

[7] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy*. IEEE, 2017. 1, 2, 6, 7, 8, 13, 14, 15

[8] Yair Carmon, Aditi Raghunathan, Ludwig Schmidt, John C Duchi, and Percy S Liang. Unlabeled data improves adversarial robustness. In *Advances in Neural Information Processing Systems*, 2019. 6, 14, 15, 17, 18, 19

[9] Pin-Yu Chen, Yash Sharma, Huan Zhang, Jinfeng Yi, and Cho-Jui Hsieh. Ead: Elastic-net attacks to deep neural networks via adversarial examples. In *Association for the Advancement of Artificial Intelligence*, 2018. 1, 6, 7, 8, 13, 14, 15

[10] A Conn, Nick Gould, and Ph Toint. A globally convergent lagrangian barrier algorithm for optimization with general inequality constraints and simple bounds. *Mathematics of Computation*, 66(217), 1997. 2, 3

[11] 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. 6

[12] Francesco Croce and Matthias Hein. Minimally distorted adversarial examples with a fast adaptive boundary attack. In *International Conference on Machine Learning*, 2020. 1, 6, 7, 8, 13, 14, 15

[13] Francesco Croce and Matthias Hein. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *International Conference on Machine Learning*, 2020. 2, 6, 7, 8, 11, 13, 14, 15

[14] Logan Engstrom, Andrew Ilyas, Hadi Salman, Shibani Santurkar, and Dimitris Tsipras. Robustness (python library), 2019. 6, 7

[15] Roger Fletcher. *Practical Methods of Optimization*. John Wiley & Sons, 2013. 2, 3

[16] Diego Gragnaniello, Francesco Marra, Giovanni Poggi, and Luisa Verdoliva. Perceptual quality-preserving black-box attack against deep learning image classifiers. *arXiv preprint arXiv:1902.07776*, 2019. 1, 12

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016. 6

[18] Paul A Jensen, Jonathan F Bard, and Patsy Jensen. *Operations Research Models and Methods*. John Wiley & Sons, 2003. 1

[19] Hoel Kervadec, Jose Dolz, Jing Yuan, Christian Desrosiers, Eric Granger, and Ismail Ben Ayed. Constrained deep networks: Lagrangian optimization via log-barrier extensions. *arXiv preprint arXiv:1904.04205*, 2019. 1

[20] Barry W Kort and Dimitri P Bertsekas. Combined primal–dual and penalty methods for convex programming. *SIAM Journal on Control and Optimization*, 14(2), 1976. 5

[21] Cassidy Laidlaw, Sahil Singla, and Soheil Feizi. Perceptual adversarial robustness: Defense against unseen threat models. In *International Conference on Learning Representations*, 2021. 1, 3, 7, 8, 15, 16

[22] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: A simple and accurate method to fool deep neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016. 1

[23] Yurii E Nesterov. A method for solving the convex programming problem with convergence rate o (1/kˆ 2). In *Dokl. akad. nauk Sssr*, volume 269, 1983. 6

[24] Jorge Nocedal and Stephen Wright. *Numerical Optimization*. Springer Science & Business Media, 2006. 2, 3

[25] Maura Pintor, Fabio Roli, Wieland Brendel, and Battista Biggio. Fast minimum-norm adversarial attacks through adaptive norm constraints. *arXiv preprint arXiv:2102.12827*, 2021. 1, 6, 7, 8, 13, 14, 15

[26] Jérôme Rony, Luiz G Hafemann, Luiz S Oliveira, Ismail Ben Ayed, Robert Sabourin, and Eric Granger. Decoupling direction and norm for efficient gradient-based l2 adversarial attacks and defenses. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2019. 1, 6, 7, 8, 13, 14, 15

[27] Gaurav Sharma, Wencheng Wu, and Edul N Dalal. The ciede2000 color-difference formula: Implementation notes, supplementary test data, and mathematical observations. *Color Research & Application: Endorsed by Inter-Society Color Council, The Colour Group (Great Britain), Canadian Society for Color, Color Science Association of Japan, Dutch Society for the Study of Color, The Swedish Colour Centre Foundation, Colour Society of Australia, Centre Français de la Couleur*, 30(1), 2005. 3, 8, 11

[28] Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop, coursera: Neural networks for machine learning. *University of Toronto, Technical Report*, 2012. 6

[29] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4), 2004. 12

[30] Eric Wong, Frank Schmidt, and Zico Kolter. Wasserstein adversarial examples via projected sinkhorn iterations. In *International Conference on Machine Learning*, 2019. 1

[31] Kaidi Xu, Sijia Liu, Pu Zhao, Pin-Yu Chen, Huan Zhang, Quanfu Fan, Deniz Erdogmus, Yanzhi Wang, and Xue Lin. Structured adversarial attack: Towards general implementation and better interpretability. In *International Conference on Learning Representations*, 2019. 1, 11

[32] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In Edwin R. Hancock Richard C. Wilson and William A. P. Smith, editors, *British Machine Vision Conference*. BMVA Press, September 2016. 6

[33] Huan Zhang, Hongge Chen, Chaowei Xiao, Sven Gowal, Robert Stanforth, Bo Li, Duane Boning, and Cho-Jui Hsieh. Towards stable and efficient training of verifiably robust neural networks. In *International Conference on Learning Representations*, 2019. 6

[34] Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric P. Xing, Laurent El Ghaoui, and Michael I. Jordan. Theoretically principled trade-off between robustness and accuracy. In *International Conference on Machine Learning*, 2019. 6

[35] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2018. 1, 3

[36] Pu Zhao, Kaidi Xu, Sijia Liu, Yanzhi Wang, and Xue Lin. ADMM attack: An enhanced adversarial attack for deep neural networks with undetectable distortions. In *Proceedings of the 24th Asia and South Pacific Design Automation Conference*, 2019. 11

[37] Zhengyu Zhao, Zhuoran Liu, and Martha Larson. Towards large yet imperceptible adversarial image perturbations with perceptual color distance. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2020. 1, 3, 7, 8, 14, 15