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 $\ell_p$-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 $\ell_2$ variant. Building on the work of Carlini and Wagner, several other attacks have been proposed, e.g. EAD for $\ell_1$ [9] and StrAttack [31], which generalizes EAD and introduces sparsity in the perturbations. More recently, other works have tackled other distances than the standard $\ell_p$-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 $\ell_p$-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 $\ell_2$ and $\ell_\infty$ norms, which uses a linear approximation of the model at each iteration [22]; DDN for the $\ell_2$-norm, which uses projections on the $\ell_2$-ball to decouple the direction and the norm of the perturbation [26]; and FAB for the $\ell_1$, $\ell_2$ and $\ell_\infty$, 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 $\ell_p$-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 $\ell_p$-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 $\ell_p$-norms but, in practice, the implementation leverages a trust-region solver designed for each $\ell_p$-norm specifically, which limits applicability to other distances.

Penalty methods are generally applicable, and can be used for distances other than the standard $\ell_p$-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,
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 $\ell_1$-norm, $\ell_2$-norm, CIDE2000, 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 algorithms for adversarial example generation. Our modifications to design a customized Augmented Lagrangian algorithm for adversarial example generation are crucial to achieve highly competitive performances in comparisons to state-of-the-art methods. We therefore handle it with a simple projection $P_{[0,1]}$. 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]:

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

(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]:

$$DLR^+ (z, y) = \frac{z_y - \max_{i \neq y} z_i}{z_{\pi_1} - z_{\pi_3}}$$

(4)

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

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

(5)

### 2.2. Equivalent problem

In the general case, constraint $x + \delta \in X$ is not necessarily trivial. However, in our context, this corresponds to a box constraint: $0 \leq x + \delta \leq 1$. We therefore handle it with a simple projection $P_{[0,1]}$. 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]:

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

(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]:

$$DLR^+ (z, y) = \frac{z_y - \max_{i \neq y} z_i}{z_{\pi_1} - z_{\pi_3}}$$

(4)

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

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

(5)

### 2.3. Distances

Most attacks in the literature measure the size of the perturbations in terms of $\ell_p$-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 $\ell_1$ and $\ell_2$ 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:

$$\text{minimize } g(x) \text{ subject to } h(x) < 0$$

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, $\lambda$, is chosen heuristically, and the unconstrained optimization is repeated several times with increasing values of $\lambda$, 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).

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:

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

is approximately minimized w.r.t. to $x$, 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. $\forall y \in \mathbb{R}, \forall (\rho, \mu) \in (\mathbb{R}_+^*)^2$, $\frac{\partial}{\partial \rho} P(y, \rho, \mu) \geq 0$

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

Axiom 3. If, for all $j \in \mathbb{N}$, $0 < \mu_{\text{min}} \leq \mu^{(j)} \leq \mu_{\text{max}} < \infty$, 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 $j \in \mathbb{N}$, $0 < \mu_{\text{min}} \leq \mu^{(j)} \leq \mu_{\text{max}} < \infty$, 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 $\mu$ 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. $h(x) \geq 0$), and tends to 0 if the constraint is satisfied (i.e. $h(x) < 0$). 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 $\mu$ and the penalty parameter $\rho$ are modified. The penalty multiplier $\mu$ is updated to the derivative of $P$ at the current value $\mu$ w.r.t. the constraint function:

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

(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 $\rho$ is increased based on the value of the constraint function at the current inner iteration, compared to the previous one. If the constraint function has not improved (i.e. reduced) significantly, then $\rho$ is multiplied by a fixed factor, typically between 2 and 100 [4]. With higher values of the penalty parameter $\rho$, 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 $x^{(0)}$

1: for $i \leftarrow 0$ to $N - 1$
2: Using $x^{(i)}$ as initialization, minimize (approximately):
3: $G(x) = g(x) + P(h(x), \rho^{(i)}, \mu^{(i)})$
4: $x^{(i+1)} \leftarrow $ approximate minimizer of $G$
5: $\mu^{(i+1)} \leftarrow P'(h(x^{(i+1)}), \rho^{(i)}, \mu^{(i)})$
6: if the constraint does not improve then
7: Set $\rho^{(i+1)} > \rho^{(i)}$
8: else
9: $\rho^{(i+1)} \leftarrow \rho^{(i)}$
10: end if
11: 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 $\delta = \tilde{x} - x$ and the penalty multiplier $\mu$. This means that we do not need an inner convergence criterion as we are always adapting $\mu$. This also requires to adapt the increase of $\rho$, 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 $x$, 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 $\rho$ increase.

**Require:** $D$ distance function

**Require:** Penalty function $P$, initial multiplier $\mu^{(0)}$, initial penalty parameter $\rho^{(0)}$

1: Initialize $\tilde{x}^{(0)} \leftarrow x$,
2: for $i \leftarrow 0$ to $N - 1$
3: $\tilde{x} \leftarrow f(\tilde{x}^{(i)})$
4: $d^{(i)} \leftarrow DLR^+(\tilde{x}, y)$
5: $\mu^{(i+1)} \leftarrow \nabla_d P(d^{(i)}, \rho^{(i)}, \mu^{(i)})$
6: $\rho^{(i+1)} \leftarrow \tau \rho^{(i)}$
7: if no adversarial has been found and $d$ does not decrease significantly, increase $\rho$ by a factor of $\gamma$
8: end if
9: $\rho^{(i+1)} \leftarrow \rho^{(i)}$
10: end if
11: end for
12: Return $\tilde{x}^{(i)}$ that is adversarial and has smallest $D(\tilde{x}^{(i)}, x)$

---

### 3.2.1 Penalty parameters adaptation

The most critical design choices for our attack are the adaptation of the penalty parameters $\mu$ and $\rho$.

**Penalty multiplier.** $\mu$ 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 $\mu$ 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 $\mu$, which tend to make the optimization unstable. Figure 2 shows the evolution of $\mu$ during the attack, as well as the $\ell_2$-norm of the perturbation, when attacking (with ALMA $\ell_2$) a single MNIST sample with the SmallCNN model. We can see that $\mu$ regularly spike to high values when not using EMA, which in turns increases the $\ell_2$-norm of the perturbation because the penalty term dominates. To solve this issue, we smooth the values of $\mu$ using an Exponential Moving...
Algorithm 2 increases when using an EMA with $\alpha = 0.9$. Then, the evolution of the penalty multiplier $\mu$ is smoother, with a much smaller spike at the beginning and then stabilizing. As a consequence, the $\ell_2$-norm of the final adversarial perturbation is smaller when using an EMA at 2.13 without EMA and 1.71 with EMA ($\alpha = 0.9$).

![Figure 2: Example of the evolution of the penalty multiplier $\mu$ and the $\ell_2$-norm of the perturbation during the optimization when attacking a single MNIST sample for the SmallCNN model. This leads to a $\ell_2$-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 $\mu$, meaning that the perturbed sample $\hat{x}^{(i)}$ is not adversarial.](image)

Average (EMA) of $\mu$:

$$
\hat{\mu} = P'(g(x), \rho^{(i)}, \mu^{(i)}) \\
\mu^{(i+1)} = \alpha \mu^{(i)} + (1 - \alpha) \hat{\mu}
$$

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 $\mu$ is smoother, with a much smaller spike at the beginning and then stabilizing. As a consequence, the $\ell_2$-norm of the final perturbation is smaller when using an EMA at 2.17 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 $\mu$ on the safeguarding interval $[\mu_{\text{min}}, \mu_{\text{max}}] \subset \mathbb{R}_+$.

Penalty parameter. $\rho$ 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 $\rho$. 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:

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

where $y$ represents the value of the constraint: $\text{DLR}^+(z, y)$ in our case (or $\text{tDLR}^+(z, y)$ 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 $\mu$ in each iteration. This, in turn, results in an increase of $\rho$ because no adversarial example is found (see step 10 and 11 of Algorithm 2). Generally, increasing $\rho$ 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 $\rho$ 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 $\mu$ 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 $x$, such that the first gradient step (step 9 in Algorithm 2) increases $D$ by $\epsilon$, or formally, such that: $D(\hat{x}^{(1)}, x) = \epsilon$. To obtain that value, we compute the gradient $g$ of $\text{DLR}^+$ w.r.t. $x$, and find the scalar $\eta^{(0)}$ such that $D(P_{[0,1]}[x - \eta^{(0)}g], x) = \epsilon$, 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 problems. 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 $\ell_2$-adversarially trained model and a $\ell_\infty$-adversarially trained model from [34] we call SmallCNN-TRADES. To vary architectures, we also test on the $\ell_\infty$-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 $\ell_\infty$-adversarially trained Wide ResNet 28-10 from [8] and a $\ell_2$-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 ($\ell_2$ and $\ell_\infty$), 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 $\alpha$ (i.e., the coefficient of the EMA) per budget and one $\epsilon$ (i.e., the increase of $D$ in the first iteration, see Section 3.2.3) per distance. $\alpha$ and $\epsilon$ are shared across 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 $\epsilon$, 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 $\epsilon$ can quickly be found when experimenting with a new distance function. Appendix F gives the initial values of $\epsilon$ used in our experiments for each distance. The other hyperparameters are fixed. We use $\mu^{(0)} = \rho^{(0)} = 1$, $\mu_{\min} = 10^{-6}$ and $\mu_{\max} = 10^{12}$, $\gamma = 1.2$, $\tau = 0.95$ 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., $\mu$ and $\rho$) more frequently, $\gamma$ can have a smaller value.

#### Attacks.

We compare our attack with various state-of-the-art attacks from the literature. Specifically, we evaluate the $\ell_1$ variant of our attack against the EAD attack [9] with budgets of $9 \times 100$ and $9 \times 1000$, and FAB $\ell_1$ [12] and FMN $\ell_1$ [25] attacks with budgets of 100 and 1000 iterations. For the $\ell_2$ variant of our attack, we compare it to the C&W $\ell_2$ [7] attack with budgets of $9 \times 1000$ and $9 \times 10000$, DDN [26], FAB $\ell_2$ [12] and FMN $\ell_2$ [25] all with budgets of 100 and 1000 iterations.\(^1\) For FAB $\ell_1$ and $\ell_2$ 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 $\ell_1$ and $\ell_2$ 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].
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 APGD$_{DLR}$ $\ell_2$ [13] to the $\ell_2$ comparison with a binary search. For the binary search, 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 $\ell_2$-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 $\ell_2$-norm with the corresponding target distance and use a budget of $9 \times 10000$ which is enough to get a good performance while keeping an acceptable budget (compared to $9 \times 1000$).

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 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 $\ell_1$ and $\ell_2$ attacks. For ImageNet, we show the curves for ResNet-50 regularly and $\ell_2$-adversarially trained from [14] against $\ell_1$ attacks.

Across all datasets and models, our attack consistently obtains results competitive with attacks tailored to each distance. On MNIST, ALMA $\ell_1$ is the only attack to reach an ASR of 100% with only FAB $\ell_1$ outperforming it on the SmallCNN model. FMN $\ell_1$ reaches a lower median $\ell_1$ at the cost of a reduced ASR on the CROWN IBP model. ALMA $\ell_2$ is only marginally outperformed by APGD$_{DLR}$ $\ell_2$ on the SmallCNN and SmallCNN-DDN models, but again, ALMA is the only attack to consistently reach 100% ASR.
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 \( \ell_1 \) obtains worse median norms than FAB \( \ell_1 \) and FMN \( \ell_1 \), but again, at the cost of a reduced ASR for both attacks. Surprisingly, FMN \( \ell_1 \) obtains lower ASR for the higher budget variant, but with a lower median. For the \( \ell_2 \) attacks, several perform similarly, with only APGD\(_{\text{FLR}}\) \( \ell_2 \) reaching a lower \( \ell_2 \) 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 \( \epsilon \), which is distance specific, and \( \alpha \) 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


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


7746


[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