Fast Minimum-norm Adversarial Attacks through Adaptive Norm Constraints

Maura Pintor  
University of Cagliari, Italy  
Pluribus One, Italy  
maura.pintor@unica.it

Fabio Roli  
University of Cagliari, Italy  
Pluribus One, Italy  
roli@unica.it

Wieland Brendel  
Tübingen AI Center,  
University of Tübingen, Germany  
wieland.brendel@uni-tuebingen.de

Battista Biggio  
University of Cagliari, Italy  
Pluribus One, Italy  
battista.biggio@unica.it

Abstract

Evaluating adversarial robustness amounts to finding the minimum perturbation needed to have an input sample misclassified. The inherent complexity of the underlying optimization requires current gradient-based attacks to be carefully tuned, initialized, and possibly executed for many computationally-demanding iterations, even if specialized to a given perturbation model. In this work, we overcome these limitations by proposing a fast minimum-norm (FMN) attack that works with different $\ell_p$-norm perturbation models ($p = 0, 1, 2, \infty$), is robust to hyperparameter choices, does not require adversarial starting points, and converges within few lightweight steps. It works by iteratively finding the sample misclassified with maximum confidence within an $\ell_p$-norm constraint of size $\epsilon$, while adapting $\epsilon$ to minimize the distance of the current sample to the decision boundary. Extensive experiments show that FMN significantly outperforms existing $\ell_0$, $\ell_1$, and $\ell_\infty$-norm attacks in terms of perturbation size, convergence speed and computation time, while reporting comparable performances with state-of-the-art $\ell_2$-norm attacks. Our open-source code is available at: https://github.com/pralab/Fast-Minimum-Norm-FMN-Attack.

1 Introduction

Learning algorithms are vulnerable to adversarial examples, i.e., intentionally-perturbed inputs aimed to mislead classification at test time [24][3]. While adversarial examples have received much attention, evaluating the robustness of deep networks against them remains a challenge. Adversarial attacks solve a non-convex optimization problem and are thus prone to finding suboptimal solutions; in particular, all attacks make certain assumptions about the underlying geometry and properties of the optimization problem which, if violated, can derail the attack and may lead to premature conclusions regarding model robustness. That is why the vast majority of defenses published in recent years have later shown to be ineffective against more powerful white-box attacks [5][1]. Having an arsenal of diverse attacks that can be adapted to specific defenses is one of the most promising avenues for increasing confidence in white-box robustness evaluations [6][25]. While it may seem that the number of attacks is already large, most of them are just small variations of the same technique, make similar underlying assumptions and thus tend to fail jointly (see, e.g., [25], in which projected-gradient attacks all fail similarly against the “Ensemble Diversity” defense).
In this work, we focus on minimum-norm attacks for evaluating adversarial robustness, i.e., attacks that aim to mislead classification by finding the smallest input perturbation according to a given norm. In contrast to maximum-confidence attacks, which maximize confidence in a wrong class within a given perturbation budget, the former are better suited to evaluate adversarial robustness as one can compute the accuracy of a classifier under attack for any perturbation budget without re-running the attack. Within the class of gradient-based minimum-norm attacks, there are three main sub-categories: (i) soft-constraint attacks, (ii) boundary attacks and (iii) projected-gradient attacks. Soft-constraint attacks like CW \[5\] optimize a trade-off between confidence of the misclassified samples and perturbation size. This class of attacks needs a sample-wise tuning of the trade-off hyperparameter to find the smallest possible perturbation, thus requiring many steps to converge. Boundary attacks like BB \[4\] and FAB \[10\] move along the decision boundary towards the closest point to the input sample. These attacks converge within relatively few steps, but BB requires an adversarial starting point, and both attacks need to solve a relatively expensive optimization problem in each step. Finally, recent minimum-norm projected-gradient attacks like DDN \[23\] perform a maximum-confidence attack in each step under a given perturbation budget $\epsilon$, while iteratively adjusting $\epsilon$ to reduce the perturbation size. DDN combines the effectiveness of boundary attacks with the simplicity and per-step speed of soft-constraint attacks; however, it is specific to the $\ell_2$ norm and cannot be readily extended to other norms.

To overcome the aforementioned limitations, in this work we propose a novel, fast minimum-norm (FMN) attack (Sect. 2), which retains the main advantages of DDN while generalizing it to different $\ell_p$ norms ($p = 0, 1, 2, \infty$). We perform large-scale experiments on different datasets and models (Sect. 3), showing that FMN is able to significantly outperform current minimum-norm attacks in terms of convergence speed and computation time (except for $\ell_2$-norm attacks, for which FMN achieves comparable results), while finding equal or better optima, on average, across almost all tested scenarios and $\ell_p$ norms. FMN thus combines all desirable traits a good adversarial attack should have, providing an important step towards improving adversarial robustness evaluations. We conclude the paper by discussing related work (Sect. 4) and future research directions (Sect. 5).

2 Minimum-Norm Adversarial Examples with Adaptive Projections

Problem formulation. Given an input sample $x \in [0, 1]^d$, belonging to class $y \in \{1, \ldots, c\}$, the goal of an untargeted attack is to find the minimum-norm perturbation $\delta^*$ such that the corresponding adversarial example $x^* = x + \delta^*$ is misclassified. This problem can be formulated as:

$$\delta^* \in \arg \min_{\delta} \|\delta\|_p,$$

s.t. $L(x + \delta, y, \theta) < 0$,  
$x + \delta \in [0, 1]^d$,  

Figure 1: (a) Conceptual representation of the FMN attack algorithm (leftmost plot). The $\epsilon$-step updates the constraint size $\epsilon$ to minimize its distance to the boundary. The $\delta$-step updates the perturbation $\delta$ with a projected-gradient step to maximize misclassification confidence within the current $\epsilon$-sized constraint. (b) Example of execution of our attack on a bi-dimensional problem (middle plot), along with the corresponding values of the loss function $L$ and the constraint size $\epsilon$ across iterations (rightmost plot). Our algorithm works by first pushing the initial point (red dot) towards the adversarial region (in red), and then perturbing it around the decision boundary to improve the current solution towards a local optimum. The vertical lines in the rightmost plot highlight the steps in which a better solution ($\text{smaller } \|\delta^*\| \text{ and } L < 0$) is found.
Algorithm 1 Fast Minimum-norm (FMN) Attack

Input: $x$, the input sample; $t$, a variable denoting whether the attack is targeted ($t = +1$) or untargeted ($t = -1$); $y$, the target (true) class label if the attack is targeted (untargeted); $\gamma_0$ and $\gamma_K$, the initial and final $\epsilon$-step sizes; $\alpha_0$ and $\alpha_K$, the initial and final $\delta$-step sizes; $K$, the total number of iterations.

Output: The minimum-norm adversarial example $x^*$.

1: $x_0 \leftarrow x$, $\epsilon_0 = 0$, $\delta_0 \leftarrow 0$, $\delta^* \leftarrow \infty$
2: for $k = 1, \ldots, K$ do
3: \[ g \leftarrow t \cdot \nabla \delta L(x_{k-1} + \delta, y, \theta) \] // loss gradient
4: \[ \gamma_k \leftarrow h(\gamma_0, \gamma_k, k) \] // $\gamma$-step size decay (Eq. 6)
5: if $L(x_{k-1}, y, \theta) \geq 0$ then
6: \[ \epsilon_k = \|\delta_{k-1}\|_p + L(x_{k-1}, y, \theta)/\|g\|_q \] if adversarial not found yet else $\epsilon_k = \epsilon_{k-1}(1 + \gamma_k)$
7: else
8: if $\|\delta_{k-1}\|_p \leq \|\delta^*\|_p$ then
9: $\delta^* \leftarrow \delta_{k-1}$ // update best min-norm solution
10: end if
11: $\epsilon_k = \min(\epsilon_{k-1}(1 - \gamma_k), \|\delta^*\|_p)$
12: end if
13: $\alpha_k \leftarrow h(\alpha_0, \alpha_K, k, K)$ // $\alpha$-step size decay (Eq. 6)
14: $\delta_{k} \leftarrow \delta_{k-1} + \alpha_k \cdot g/\|g\|_2$ // gradient-scaling step
15: $\delta_{k} \leftarrow \Pi_{\|\cdot\|_2}(x_0 + \delta_{k}) - x_0$
16: $\delta_{k} \leftarrow \text{clip}(x_0 + \delta_{k}) - x_0$
17: $x_k \leftarrow x_0 + \delta_{k}$
18: end for
19: return $x^* \leftarrow x_0 + \delta^*$

where $\| \cdot \|_p$ indicates the $\ell_p$-norm operator. The loss $L$ in the constraint in Eq. (2) is defined as:

\[ L(x, y, \theta) = f_y(x, \theta) - \max_{j \neq y} f_j(x, \theta), \] (4)

where $f_j(x, \theta)$ is the confidence given by the model $f$ for classifying $x$ as class $j$, and $\theta$ is the set of its learned parameters. Assuming that the classifier assigns $x$ to the class exhibiting the highest confidence, i.e., $y^* = \arg \max_{j \in 1, \ldots, c} f_j(x, \theta)$, the loss function $L(x, y, \theta)$ takes on negative values only when $x$ is misclassified. Finally, the box constraint in Eq. (3) ensures that the perturbed sample $x + \delta$ lies in the feasible input space. The aforementioned problem typically involves a non-convex loss function $L$ (w.r.t. its first argument), due to the non-convexity of the underlying decision function $f$. For this reason, it may admit different locally-optimal solutions. Note also that the solution is trivial (i.e., $\delta^* = 0$) when the input sample $x$ is already adversarial (i.e., $L(x, y, \theta) < 0$).

Extension to the targeted case. The goal of a targeted attack is to have the input sample misclassified in a given target class $y'$. This can be accounted for by modifying the loss function in Eq. (4) as

\[ L'(x, y', \theta) = \max_{y \neq y'} f_j(x, \theta) + f_{y'}(x, \theta) - f_{y}(x, \theta) = -L(x, y', \theta), \] i.e., changing its sign and using the target class label $y'$ instead of the true class label $y$.

Solution algorithm. To solve Problem (1)-(3), we reformulate it using an upper bound $\epsilon$ on $\|\delta\|_p$:

\[ \min_{\epsilon, \delta} \epsilon, \quad \text{s.t. } \|\delta\|_p \leq \epsilon, \] (5)

and to the constraints in Eqs. (2)-(3). This allows us to derive an algorithm that works in two main steps, similarly to DDN [23], by updating the maximum perturbation size $\epsilon$ separately from the actual perturbation $\delta$, as represented in Fig. 1(a). In particular, the constraint size $\epsilon$ is adapted to reduce the distance of the constraint to the boundary ($\epsilon$-step), while the perturbation $\delta$ is updated using a projected-gradient step to minimize the loss function $L$ within the given $\epsilon$-sized constraint ($\delta$-step). This essentially amounts to a projected gradient descent algorithm that iteratively adapts the constraint size $\epsilon$ to find the minimum-norm adversarial example. The complete algorithm is given as Algorithm 1 while a more detailed explanation of the two aforementioned steps is given below.

$\epsilon$-step. This step updates the upper bound $\epsilon$ on the perturbation norm (lines 12-18 in Algorithm 1). The underlying idea is to increase $\epsilon$ if the current sample is not adversarial (i.e., $L(x_{k-1}, y, \theta) \geq 0$), and to decrease it otherwise, while reducing the step size to dampen oscillations around the boundary.
and reach convergence. In the former case ($\epsilon$-increase), the increment of $\epsilon$ depends on whether an adversarial example has been previously found or not. If not, we estimate the distance to the boundary with a first-order linear approximation, and set $\epsilon_k = ||\delta_{k-1}||_p + L(x_{k-1}, y, \theta)/||\nabla L(x_{k-1}, y, \theta)||_q$, where $q$ is the dual norm of $p$. This approximation allows the attack point to make faster progress towards the decision boundary. Conversely, if an adversarial sample has been previously found, but the current sample is not adversarial, it is likely that the current estimate of $\epsilon$ is only slightly smaller than the minimum-norm solution. We thus increase $\epsilon$ by a small fraction as $\epsilon_k = \epsilon_{k-1} (1 + \gamma_k)$, being $\gamma_k$ a decaying step size. In the latter case ($\epsilon$-decrease), if the current sample is adversarial, i.e., $L(x_{k-1}, y, \theta) < 0$, we decrease $\epsilon$ as $\epsilon_k = \epsilon_{k-1} (1 - \gamma_k)$, to check whether the current solution can be improved. If the corresponding $\epsilon_k$ value is larger than the optimal $||\delta^*||_p$ found so far, we retain the best value and set $\epsilon_k = ||\delta^*||_p$. These multiplicative updates of $\epsilon$ exhibit an oscillating behavior around the decision boundary, due to the conflicting requirements of minimizing the perturbation size and finding an adversarial point. To ensure convergence, as anticipated before, the step size $\gamma_k$ is decayed with cosine annealing:

$$
\gamma_k = h(\gamma_0, \gamma_K, k, K) = \gamma_K + \frac{1}{2}(\gamma_0 - \gamma_K) \left(1 + \cos \left(\frac{k \pi}{K}\right)\right),
$$

being $k$ the current step, $K$ the total number of steps, and $\gamma_0$ and $\gamma_K$ the initial and final step sizes.

$\delta$-step. This step updates $\delta$ (lines [3]–[7] in Algorithm 1). The goal is to find the adversarial example that is misclassified with maximum confidence (i.e., for which $L$ is minimized) within the current $\epsilon$-sized constraint (Eq. 5) and bounds (Eq. 3). This amounts to performing a projected-gradient step along the negative gradient of $L$. We consider a normalized steepest descent with decaying step size $\alpha$ to overcome potential issues related to noisy gradients while ensuring convergence (line 14). Note that this step only rescales the gradient by its $\ell_2$ norm, while preserving its direction. The step size $\alpha$ is decayed using cosine annealing (Eq. 6). Once $\delta$ is updated, we project it onto the given $\epsilon$-sized $\ell_p$-norm constraint via a projection operator $\Pi_{\epsilon}$ (line 15), to fulfill the constraint in Eq. 5. The projection is trivial for $p = \infty$ and $p = 2$. For $p = 1$, we use the efficient algorithm by Duchi et al. [14]. For $p = 0$, we retain only the first $\epsilon$ components of $\delta$ exhibiting the largest absolute value. We finally clip the components of $\delta$ that violate the bounds in Eq. 5 (line 16).

Execution example. In Fig. 1(b), we report an example of execution of our algorithm on a bi-dimensional problem. The initial sample is updated to follow the negative gradient of $L$ towards the decision boundary. When an adversarial point is found, the algorithm reduces $\epsilon$ to find a better solution. The point is thus projected back onto the non-adversarial region, and $\epsilon$ increased (by a smaller, decaying amount). These oscillations allow the point to walk on the boundary towards a local optimum, i.e., an adversarial point lying on the boundary, where the gradient of the loss function and that of the norm constraint have opposite direction. FMN tends to quickly converge to a good local optimum, provided that the step size is reduced to a sufficiently-small value and that a sufficiently-large number of iterations are performed. This is also confirmed empirically in Sect. 3.

Adversarial initialization. Our attack can be initialized from the input sample $x$, or from a point $x_{\text{init}}$ belonging either to a different class (if the attack is untargeted) or to the target class (if the attack is targeted). When initializing the attack from $x_{\text{init}}$, we perform a 10-step binary search between $x$ and $x_{\text{init}}$, to find an adversarial point which is closer to the decision boundary. In particular, we aim to find the minimum $\epsilon$ such that $L(x + \Pi_{\epsilon}(x_{\text{init}} - x), y, \theta) < 0$ (or $L^t < 0$ for targeted attacks). Then we run our attack starting from the corresponding values of $x_k$, $\epsilon_k$, $\delta_k$ and $\delta^*$. Differences with DDN. FMN applies substantial changes to both the algorithm and the formulation of DDN. The main difference is that (i) DDN always rescales the perturbation to have size $\epsilon$. This operation is problematic when using other norms, especially sparse ones, as it hinders the ability of the attack to explore the neighboring space and find a suitable descent direction. Another difference is that (ii) FMN does not use the cross-entropy loss, but it uses the logit difference as the loss function $L$, since the latter is less affected by saturation effects. Moreover, (iii) FMN does not need an initial value for $\epsilon$, as $\epsilon$ is dynamically estimated; and (iv) $\gamma$ is decayed to improve convergence around better minimum-norm solutions, by more effectively dampening oscillations around the boundary. Finally, we include the possibility of (v) initializing the attack from an adversarial point, which can greatly increase the convergence speed of the algorithm, as it uses a fast line-search algorithm to find the boundary and the remaining queries to refine the result.
3 Experiments

We report here an extensive experimental analysis involving several state-of-the-art defenses and minimum-norm attacks, covering $\ell_0$, $\ell_1$, $\ell_2$ and $\ell_\infty$ norms. The goal is to empirically benchmark our attack and assess its effectiveness and efficiency as a tool for adversarial robustness evaluation.

3.1 Experimental Setup

**Datasets.** We consider two commonly-used datasets for benchmarking adversarial robustness of deep neural networks, i.e., the MNIST handwritten digits and CIFAR10. Following the experimental setup in [4], we use a subset of 1000 test samples to evaluate the considered attacks and defenses.

**Models.** We use a diverse selection of models to thoroughly evaluate attacks under different conditions. For MNIST, we consider the following four models: $M1$, the 9-layer network used as the undefended baseline model by Papernot et al. [20]; Carlini and Wagner [5]; $M2$, the robust model by Madry et al. [17], trained on $\ell_\infty$ attacks (robustness claim: 89.6% accuracy with $||\delta||_\infty \leq 0.3$, current best evaluation: 88.0%); $M3$, the robust model by Rony et al. [23], trained on $\ell_2$ attacks (robustness claim: 87.6% accuracy with $||\delta||_2 \leq 1.5$); and $M4$, the IBP Large Model by Zhang et al. [27] (robustness claim: 94.3% accuracy with $||\delta||_\infty \leq 0.3$). For CIFAR10, we consider three state-of-the-art robust models from RobustBench [11]: $C1$, the robust model by Madry et al. [17], trained on $\ell_\infty$ attacks (robustness claim: 44.7% accuracy with $||\delta||_\infty \leq 8/255$, current best evaluation: 44.0%); $C2$, the defended model by Carmon et al. [7] (top-5 in RobustBench), trained on $\ell_\infty$ attacks and additional unsupervised data (robustness claim: 62.5% accuracy with $||\delta||_\infty \leq 8/255$, current best evaluation: 59.5%); and $C3$, the robust model by Rony et al. [23], trained on $\ell_2$ attacks (robustness claim: 67.9% accuracy with $||\delta||_2 \leq 0.5$, current best evaluation: 66.4%).

**Attacks.** We compare our algorithm against different state-of-the-art attacks for finding minimum-norm adversarial perturbations across different norms: the Carlini & Wagner (CW) attack [5], the Decoupling Direction and Norm (DDN) attack [23], the Brendel & Bethge (BB) attack [4], and the Fast Adaptive Boundary (FAB) attack [10]. We use the implementation of FAB from Ding et al. [12], while for all the remaining attacks we use the implementation available in Foolbox [21][22]. All these attacks are defined on the $\ell_2$ norm. BB and FAB are also defined on the $\ell_1$ and $\ell_\infty$ norms, and only BB is defined on the $\ell_0$ norm. We consider both untargeted and targeted attack scenarios, as defined in Sect. 2 except for FAB, which is only evaluated in the untargeted case [5].

**Hyperparameters.** To ensure a fair comparison, we perform an extensive hyperparameter search for each of the considered attacks. We consider two main scenarios: tuning the hyperparameters at the sample-level and at the dataset-level. In the sample-level scenario, we select the optimal hyperparameters separately for each input sample by running each attack 10 to 16 times per sample, with a different hyperparameter configuration or random initialization point each time. In the dataset-level scenario, we choose the same hyperparameters for all samples, selecting the configuration that yields the best attack performance. While sample-level tuning provides a fairer comparison across attacks, it is more computationally demanding and less practical than dataset-level tuning. In addition, the latter allows us to understand how robust attacks are to suboptimal hyperparameter choices. We select the hyperparameters to be optimized for each attack as recommended by the corresponding authors [4][5][10][23]. The hyperparameter configurations considered for each attack are detailed below. For attacks that are claimed to be robust to hyperparameter changes, like BB and FAB, we follow the recommendation of using a larger number of random restarts rather than increasing the number of hyperparameter configurations to be tested. In addition, as BB requires being initialized from an adversarial starting point, we initialize it by randomly selecting a sample either from a different class (in the untargeted case) or from the target class (in the targeted case). Finally, as each attack performs operations with different levels of complexity within each iteration, possibly querying the model multiple times, we set the number of steps for each attack such that at least 1,000 forward passes (i.e., queries) are performed. This ensures a fairer comparison also in terms of the computational time and resources required to execute each attack.

---

1The reason is that FAB does not support generating adversarial examples with a particular target label. The targeted version of FAB aims to find a closer untargeted misclassification by running the attack a number of times, each time targeting a different candidate class, and then selecting the best solution [10][9].
We run FMN for $K = 1000$ steps, using $\gamma_0 \in \{0.05, 0.3\}$, $\gamma_K = 10^{-4}$, and $\alpha_K = 10^{-5}$. For $\ell_0$, $\ell_1$, and $\ell_2$, we set $\alpha_0 \in \{1, 5, 10\}$. For $\ell_\infty$, we set $\alpha_0 \in \{10^1, 10^2, 10^3\}$, as the normalized $\ell_2$ step yields much smaller updates in the $\ell_\infty$ norm. For each hyperparameter setting we run the attack twice, starting from (i) the input sample and (ii) an adversarial point.

**Evaluation criteria.** We evaluate the attacks along four different criteria: (i) perturbation size and (ii) robustness to hyperparameter selection, measured as the median $\|\delta\|_p$ on the test set (for a fixed budget of $Q$ queries and for sample- and dataset-level hyperparameter tuning, where by “robustness” we mean that a fixed hyperparameter configuration works well across different samples); (iii) execution time, measured as the average time spent per query (in milliseconds); and (iv) convergence speed, measured as the average number of queries required to converge to a good-enough solution (within 10% of the best value found at $Q = 1000$). When computing the median, we follow the evaluation in [4]: the perturbation size is set to 0 if a clean sample is misclassified, while it is set
to $\infty$ when the attack fails (no adversarial is found). The median perturbation size thus represents the value for which 50% of the samples evade a particular model.

3.2 Experimental Results

Query-distortion (QD) curves. To evaluate each attack in terms of perturbation size under the same query budget $Q$, we use the so-called QD curves introduced by Brendel et al. [4]. These curves report, for each attack, the median value of $\delta^*$ as a function of the number of queries $Q$. For each given $Q$ value, the optimal $\delta^*$ for each point is selected among the different attack executions (i.e., using different hyperparameters and/or initialization points, as described in Sect. 3.1). In Fig. 2 we report the QD curves for the MNIST and CIFAR10 challenge models (i.e., M2 and C1) in the untargeted scenario. The remaining QD curves exhibit a similar behavior and can be found in the supplementary material. It is worth noting that our attack attains comparable results in terms of perturbation size across all norms, while significantly outperforming FAB and BB in the $\ell_1$ case. It typically requires also less iterations than the other attacks to converge. While the QD curves show the complete behavior of each attack as $Q$ increases, a more compact and thorough summary of our evaluation is reported below, according to the four evaluation criteria described in Sect. 3.1.

Perturbation size. Table 1 reports the median value of $\|\delta^*\|$ at $Q = 1000$ queries (i.e., the last value from the query-distortion curve), for all models, attacks and norms. The values obtained with sample-level hyperparameter tuning confirm that our attack can find smaller or comparable perturbations with those found by the competing attacks, in most of the untargeted and targeted cases, and that the biggest margin is achieved in the $\ell_1$ case. FMN is only slightly worse than DDN and BB in a few cases, including $\ell_2$-DDN on M4 and $\ell_\infty$-BB on M2 and M4. The reason may be that these robust models exhibit noisy gradients and flat regions around the clean input samples, hindering the initial optimization steps of the FMN attack.

Robustness to hyperparameter selection. The values reported in the lower part of Table 1 show that, when using dataset-level hyperparameter tuning, FMN outperforms the other attacks in a much larger number of cases. This shows that FMN is more robust to hyperparameter changes, while other attacks like $\ell_0$- and $\ell_1$-BB suffer when using the same hyperparameters for all samples.

Execution time. The average runtime per query for each attack-model pair, measured on a workstation with an NVIDIA GeForce RTX 2080 Ti GPU with 11GB of RAM, can be found in Table 2. The results show that our attack is up to 2-3 times faster, with the exception of DDN in the $\ell_2$ case. This is however compensated by the fact that FMN finds better solutions. The advantage is that our attack avoids costly inner projections as in BB and FAB. FMN is slightly less time-efficient than DDN and CW, as it simultaneously updates the adversarial point and the norm constraint. In particular, the update on the constraint may initially require computing the norm of the gradient $g$ (line 5 in Algorithm 1), which increases the runtime of our attack. FAB computes a similar step, but for all the output classes, which hinders its scalability to problems with many classes.

Convergence speed. To get an estimate of the convergence speed, we measure the number of queries required by each attack to reach a perturbation size that is within 10% of the value found at $Q = 1000$ queries (the lower the better). Results are shown in Table 3. Our attack converges on par with or faster than all other attacks for almost all models, often requiring only half or a fifth as many queries as the state of the art. Exceptions are MNIST and CIFAR10 challenge models (M2 and C1) for $\ell_2$ and $\ell_\infty$, where BB and DDN occasionally converge faster. FMN rarely needs more than 100 steps, reaching the minimal perturbation after only 10-30 queries on many datasets, models and norms.

Robust accuracy. Despite our attack being not tailored to target specific defenses, and our evaluation restricted to a subset of the testing samples, it is worth remarking that the robust accuracies of the models against our attack are aligned with that reported in current evaluations, with the notable exception of C3, where our attack can decrease robust accuracy from 67.9% to 65.5%.

Experiments on ImageNet. We conclude our experiments by running an additional comparison between FMN and a widely-used maximum-confidence attack, i.e., the Projected Gradient Descent (PGD) attack [12], on two pretrained ImageNet models (i.e., ResNet18 and VGG16), considering $\ell_1$, $\ell_2$ and $\ell_\infty$ norms. The hyperparameters are tuned at the dataset-level using 20 validation samples. For FMN, we fix the hyperparameters as discussed before, and only tune $\alpha_0 \in \{0.1, 1, 2, 8\}$, without using adversarial initialization. For PGD, we tune the step size $\alpha \in \{0.001, 0.01, 0.1, 1, 2, 8\}$. We run both attacks for $Q = 1,000$ queries on a separate set of 1,000 samples. The success rates of both
attacks at fixed $\epsilon$ values are reported in Table 4. The results show that FMN outperforms or equals PGD in all norms.

4 Related Work

Gradient-based attacks on machine learning have a long history [3, 2]. Maximum-confidence attacks optimize the adversarial loss (e.g., the difference between the logits of the true class and the best non-true class) to find an adversarial point misclassified with maximum confidence within a given, bounded perturbation size. While attacks in this category like FGSM [15], PGD [16, 17] and momentum-based extensions of PGD [26, 15] are popular, they only partially evaluate the adversarial
robustness of a model. Minimum-norm attacks aim to minimize the norm of the perturbation subject to being adversarial. Attacks from this class give a more complete picture of the model robustness and allow us to compute the accuracy of the model under attacks with any post-hoc defined minimum perturbation size. L-BFGS [24] solves this problem with a quasi-Newton optimizer while CW [5] and EAD [8] use first-order gradient-based optimizers to minimize a weighted loss between perturbation size and misclassification confidence. To find the smallest adversarial perturbation, both CW and EAD need to tune the relative weighting which makes them query-inefficient. DeepFool [19] and SparseFool [18] compute gradients with respect to all classes in each step to estimate a linear approximation of the model from which the optimal adversarial perturbation can be computed. These two attacks are fast but do not converge to competitive solutions. BB [4] and FAB [10] use complex projections and approximations to stay close to the decision boundary (using the gradient to estimate the local geometry of the boundary) while minimizing the norm. This way of formulating minimum-norm optimization bypasses the tuning of a weighting term, but in the case of BB it also requires an adversarial starting point to begin with. The DDN attack [23] maximizes the adversarial criterion within a given norm constraint, and iteratively reduces the norm to find the smallest possible adversarial perturbation; however, it is constrained to $\ell_2$ and does not perform well on other $\ell_p$ norms.

The proposed FMN attack belongs to the category of minimum-norm attacks, and builds on BB, FAB and DDN to retain their main advantages. First, FMN is not specific to a given norm, and converges in many fewer steps than soft-constraint attacks like CW, as it does not need to optimize the trade-off between perturbation size and misclassification confidence. FMN needs significantly less computational time per step than the other attacks, it is extremely fast, by (iii) reducing runtime up to 3 times per query with respect to competing attacks and (iv) converging within less iterations. FMN also works with different $\ell_p$ norms ($p = 0, 1, 2, \infty$) and it does not necessarily require being initialized from an adversarial starting point.

Table 4: Success rate (%) of FMN against PGD on ImageNet models.

| Model     | $\ell_0$ | $\ell_1$ | $\ell_2$ | $\ell_\infty$ |
|-----------|----------|----------|----------|---------------|
| ResNet18  | 31.4     | 30.4     | 61.7     | 65.8          |
| VGG       | 38.4     | 39.8     | 61.4     | 66.2          |
| FMN       | 51.0     | 49.9     | 55.2     | 49.0          |

5 Contributions, Limitations, and Future Work

This work introduces a novel minimum-norm attack that combines all desirable traits to help improve current adversarial evaluations: (i) finding smaller or comparable minimum-norm perturbations across a range of models and datasets; (ii) being less sensitive to hyperparameter choices; and being extremely fast, by (iii) reducing runtime up to 3 times per query with respect to competing attacks and (iv) converging within less iterations. FMN also works with different $\ell_p$ norms ($p = 0, 1, 2, \infty$) and it does not necessarily require being initialized from an adversarial starting point. Our experiments have shown that FMN rivals or surpasses other attacks in speed, reliability, efficacy and versatility. While FMN is able to find smaller perturbations consistently when compared against $\ell_0$ and $\ell_1$ attacks, it only rivals the performance of other attacks for $\ell_2$ and $\ell_\infty$ norms, especially when tested against robust models which may present obfuscated gradients. To overcome this limitation, FMN may be extended using smoothing strategies that help find better descent directions, e.g., by averaging...
gradients on randomly-perturbed inputs. This can be regarded as an interesting extension of FMN towards attacking robust models. In this respect, we also believe that FMN may facilitate minimum-norm adaptive evaluations in a more general sense. Adaptive evaluations, where the attack is modified to be maximally effective against a new defense, are the key element towards properly evaluating adversarial robustness [6,25]. PGD attacks are popular in part for the ease by which they can be adapted to new defenses. Since FMN combines PGD with a dynamic minimization of the perturbation size, we argue that our attack can also be easily adapted to new defenses, thereby facilitating adaptive evaluations. FMN may also benefit from other improvements that have been suggested for PGD, including momentum, cyclical step sizes or restarts. We leave such improvements to future work.

To conclude, we firmly believe that FMN will establish itself as a useful tool in the arsenal of robustness evaluation. By facilitating more reliable robustness evaluations, we expect that FMN will foster advancements in the development of machine-learning models with improved robustness guarantees. We thus argue that there are neither ethical aspects nor evident future societal consequences with potential negative impacts that should be specifically addressed in the context of this work.

Acknowledgements

This work has been partly supported by the PRIN 2017 project RexLearn (grant no. 2017TWNMH2), funded by the Italian Ministry of Education, University and Research; by the EU H2020 project ALOHA, under the European Union’s Horizon 2020 research and innovation programme (grant no. 780788); and by BMK, BMDW, and the Province of Upper Austria in the frame of the COMET Programme managed by FFG in the COMET Module S3AI. Wieland Brendel acknowledges support from the German Federal Ministry of Education and Research (BMBF) through the Competence Center for Machine Learning (TUE.AI, FKZ 01IS18039A), from the German Science Foundation (DFG) under grant no. BR 6382/1-1 (Emmy Noether Program) as well as support by Open Philantropy and the Good Ventures Foundation.

References

[1] A. Athalye, N. Carlini, and D. A. Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In ICML, volume 80 of JMLR Workshop and Conf. Proceedings, pages 274–283. JMLR.org, 2018.

[2] B. Biggio and F. Roli. Wild patterns: Ten years after the rise of adversarial machine learning. Pattern Recognition, 84:317–331, 2018.

[3] B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Šrndić, P. Laskov, G. Giacinto, and F. Roli. Evasion attacks against machine learning at test time. In H. Blockeel, K. Kersting, S. Nijssen, and F. Železný, editors, Machine Learning and Knowledge Discovery in Databases (ECML PKDD), Part III, volume 8190 of LNCS, pages 387–402. Springer Berlin Heidelberg, 2013.

[4] W. Brendel, J. Rauber, M. Kümmerer, I. Ustyuzhaninov, and M. Bethge. Accurate, reliable and fast robustness evaluation. In Advances in Neural Inf. Proc. Systems, pages 12861–12871, 2019.

[5] N. Carlini and D. Wagner. Towards evaluating the robustness of neural networks. In 2017 ieee symposium on security and privacy (sp), pages 39–57. IEEE, 2017.

[6] N. Carlini, A. Athalye, N. Papernot, W. Brendel, J. Rauber, D. Tsipras, I. Goodfellow, A. Madry, and A. Kurakin. On evaluating adversarial robustness. arXiv preprint arXiv:1902.06705, 2019.

[7] Y. Carmon, A. Raghunathan, L. Schmidt, J. C. Duchi, and P. S. Liang. Unlabeled data improves adversarial robustness. In Advances in Neural Inf. Proc. Systems, pages 11192–11203, 2019.

[8] P.-Y. Chen, Y. Sharma, H. Zhang, J. Yi, and C.-J. Hsieh. EAD: Elastic-net attacks to deep neural networks via adversarial examples. arXiv preprint arXiv:1709.04114, 2017.

[9] F. Croce and M. Hein. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In ICML, 2020.

[10] F. Croce and M. Hein. Minimally distorted adversarial examples with a fast adaptive boundary attack. In Int’l Conf. on Machine Learning, pages 2196–2205. PMLR, 2020.
[11] F. Croce, M. Andriushchenko, V. Sehwag, N. Flammarion, M. Chiang, P. Mittal, and M. Hein. Robustbench: a standardized adversarial robustness benchmark. *arXiv preprint arXiv:2010.09670*, 2020.

[12] G. W. Ding, L. Wang, and X. Jin. AdverTorch v0.1: An adversarial robustness toolbox based on pytorch. *arXiv preprint arXiv:1902.07623*, 2019.

[13] Y. Dong, F. Liao, T. Pang, H. Su, J. Zhu, X. Hu, and J. Li. Boosting adversarial attacks with momentum. In *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition*, 2018.

[14] J. Duchi, S. Shalev-Shwartz, Y. Singer, and T. Chandra. Efficient projections onto the l 1-ball for learning in high dimensions. In *Proceedings of the 25th Int’l Conf. on Machine learning*, pages 272–279, 2008.

[15] I. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. In *Int’l Conf. on Learning Representations*, 2015. URL [http://arxiv.org/abs/1412.6572](http://arxiv.org/abs/1412.6572).

[16] A. Kurakin, I. Goodfellow, and S. Bengio. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533*, 2016.

[17] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[18] A. Modas, S.-M. Moosavi-Dezfooli, and P. Frossard. Sparsefool: a few pixels make a big difference. In *The IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[19] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard. Deepfool: A simple and accurate method to fool deep neural networks. In *The IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[20] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *2016 IEEE symposium on security and privacy (SP)*, pages 582–597. IEEE, 2016.

[21] J. Rauber, W. Brendel, and M. Bethge. Foolbox: A python toolbox to benchmark the robustness of machine learning models. In *Reliable Machine Learning in the Wild Workshop, 34th Int’l Conf. on Machine Learning*, 2017. URL [http://arxiv.org/abs/1707.04131](http://arxiv.org/abs/1707.04131).

[22] J. Rauber, R. Zimmermann, M. Bethge, and W. Brendel. Foolbox native: Fast adversarial attacks to benchmark the robustness of machine learning models in pytorch, tensorflow, and jax. *Journal of Open Source Software*, 5(53):2607, 2020. doi: 10.21105/joss.02607. URL [https://doi.org/10.21105/joss.02607](https://doi.org/10.21105/joss.02607).

[23] J. Rony, L. G. Hafemann, L. S. Oliveira, I. B. Ayed, R. Sabourin, and E. Granger. Decoupling direction and norm for efficient gradient-based l2 adversarial attacks and defenses. In *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition*, pages 4322–4330, 2019.

[24] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. In *Int’l Conf. on Learning Representations*, 2014. URL [http://arxiv.org/abs/1312.6199](http://arxiv.org/abs/1312.6199).

[25] F. Tramer, N. Carlini, W. Brendel, and A. Madry. On adaptive attacks to adversarial example defenses. In *Advances in Neural Inf. Proc. Systems*, 2020.

[26] J. Uesato, B. O’Donoghue, A. v. d. Oord, and P. Kohli. Adversarial risk and the dangers of evaluating against weak attacks. *arXiv preprint arXiv:1802.05666*, 2018.

[27] H. Zhang, H. Chen, C. Xiao, S. Gowal, R. Stanforth, B. Li, D. Boning, and C.-J. Hsieh. Towards stable and efficient training of verifiably robust neural networks. In *Int’l Conf. on Learning Representations*, 2020. URL [https://openreview.net/forum?id=Skxuk1rFwB](https://openreview.net/forum?id=Skxuk1rFwB).
Appendix

In this section we first show adversarial examples obtained by different $\ell_p$ attacks on MNIST and CIFAR10 data for visual comparison. These examples highlight the different behavior exhibited by each attack. We then report the query-distortion curves for all datasets, models and attacks used in this paper, showing that our attack outperforms current attacks on the $\ell_1$ norm and rivals their performance on other norms, while typically converging with much fewer queries.

A1. Adversarial Examples

In Figs. 3-4, we report adversarial examples generated by all attacks against model M2 and C2, respectively, on MNIST and CIFAR10 datasets, in the untargeted scenario.

The clean samples and the original label are displayed in the first row of each figure. In the remaining rows we show the perturbed sample along with the predicted class and the corresponding norm of perturbation $\|\delta^*\|_p$. It is worth noting that the output class for different untargeted attacks is not always the same, which might sometimes explain differences in the perturbation sizes. An example is given in Fig. 4b where the sample in the fourth column, labeled as “ship”, is perturbed by most of the attacks towards the class “airplane”, while in our case it outputs the class “dog” with a much smaller distance.

A2. Query-distortion Curves

In Sect. 3.2 we introduced the query-distortion curves as an efficiency evaluation metric for the attacks. We report here the complete results for all models, in targeted and untargeted scenarios.

On the MNIST dataset, our attacks generally reach smaller norms with fewer queries, with the exception of M2 (Figs. 5-6), where it seems to reach convergence more slowly than BB in $\ell_0$ and $\ell_\infty$. In $\ell_2$, the CW attack is the slowest to converge, due to the need of carefully tuning the weighting term, as described in Sect. 4.

On the CIFAR10 dataset (Figs. 7-8), our attack always rivals or outperforms the others, with the notable exception of DDN for the $\ell_2$ norm, which sometimes finds smaller perturbations more quickly, as also shown in Table 3.
(a) Untargeted $\ell_\infty$ attacks against M2 [17].  
(b) Untargeted $\ell_2$ attacks against M2 [17].  

(c) Untargeted $\ell_1$ attacks against M2 [17].  
(d) Untargeted $\ell_0$ attacks against M2 [17].

Figure 3: Adversarial examples on MNIST dataset.
Figure 4: Adversarial examples on CIFAR10 dataset.
Figure 5: Query-distortion curves for untargeted (U) attacks on the M1, M2, M3, and M4 MNIST models.
Figure 6: Query-distortion curves for targeted (T) attacks on the M1, M2, M3 and M4 MNIST models.
Figure 7: Query-distortion curves for untargeted ($U$) attacks on the C1 (top), C2 (middle), and C3 (bottom) CIFAR10 models.
Figure 8: Query-distortion curves for targeted (T) attacks on the C1 (top), C2 (middle), and C3 (bottom) CIFAR10 models.