AttackBench: Evaluating Gradient-based Attacks for Adversarial Examples

Antonio Emanuele Cinà*[1], Jérôme Rony*[2], Maura Pintor[3], Luca Demetrio[1], Ambra Demontis[3], Battista Biggio[3], Ismail Ben Ayed[2], Fabio Roli[1]

[1]University of Genoa, Italy
[2]École de Technologie Superieure, Montréal, Canada
[3]University of Cagliari, Italy
antonio.cina@unige.it, jerome.rony.1@etsmtl.net, maura.pintor@unica.it, luca.demetrio@unige.it, ambra.demontis@unica.it, battista.biggio@unica.it, ismail.benayed@etsmtl.ca, fabio.roli@unige.it

Abstract

While novel gradient-based attacks are continuously proposed to improve the optimization of adversarial examples, each is shown to outperform its predecessors using different experimental setups, implementations, and computational budgets, leading to biased and unfair comparisons. In this work, we overcome this issue by proposing AttackBench, i.e., an attack evaluation framework that evaluates the effectiveness of each attack (along with its different library implementations) under the same maximum available computational budget. To this end, we (i) define a novel *optimality* metric that quantifies how close each attack is to the optimal solution (empirically estimated by ensembling all attacks), and (ii) limit the maximum number of forward and backward queries that each attack can execute on the target model. Our extensive experimental analysis compares more than 100 attack implementations over 800 different configurations, considering both CIFAR-10 and ImageNet models, and shows that only few attack implementations outperform all the remaining approaches. These findings suggest that novel defenses should be evaluated against different attacks than those normally used in the literature to avoid overly-optimistic robustness evaluations. We release AttackBench as a publicly-available benchmark that will be continuously updated with new attack implementations to maintain an up-to-date ranking of the best gradient-based attacks.

Code — https://attackbench.github.io
Extended version — https://arxiv.org/abs/2404.19460

1 Introduction

In recent years, we have been witnessing a proliferation of gradient-based attacks, aiming to constantly improve efficacy and efficiency while optimizing adversarial examples (Carlini and Wagner 2017; Chen et al. 2018; Rony et al. 2019; Pintor et al. 2021; Cinà et al. 2024). Despite these advancements, the comparisons between each newly-proposed attack and its predecessors have been systematically conducted using different experimental settings, i.e., (i) models and performance metrics, (ii) attack implementations, and (iii) computational budgets, hindering reproducibility and an overall fair comparison of current attack algorithms. In particular, (i) many comparisons have considered a different set of target models, and used different metrics to evaluate the attack success, using either the success rate at a predefined perturbation budget (e.g., $\epsilon = 8/255$) (Croce and Hein 2019) or the median perturbation size required to mislead the model (Brendel et al. 2020; Pintor et al. 2021). However, these settings neither allow a direct comparison of the performance of different attacks across different papers nor tell anything about the effectiveness of the attacks at different perturbation budgets. Many attacks (ii) have been re-implemented without ensuring consistency with their original versions, and some implementations are even flawed (Carlini 2019). We will indeed demonstrate in this paper that different implementations of the same attack can yield significantly different results, affecting the performance of state-of-the-art methods. Attacks (iii) are not consistently compared using the same number of queries (i.e., the number of forward and backward passes performed on the target model), biasing the evaluation in favor of more computationally-demanding methods. Nevertheless, limiting the number of iterations of each attack is insufficient to solve this issue. The reason is that some attacks require two queries per iteration (one forward and one backward) (Pintor et al. 2021; Rony et al. 2019), while others use additional subroutines like restarts (Croce and Hein 2020a, 2019) or run internal hyperparameter optimizations (Chen et al. 2018; Carlini and Wagner 2017), resulting in multiple queries per iteration. These factors have led to either overestimating or underestimating the performance of certain attacks (Pintor et al. 2022; Carlini 2019). Consequently, an in-depth investigation into the effectiveness of gradient-based attacks in a fair and reproducible experimental setting is missing.

To address the above limitations, we propose AttackBench: a unified benchmark that evaluates adversarial attacks under a consistent setup and imposes a maximum computational budget for the attack (i.e., the maximum number of queries they can use). We design AttackBench as a five-stage procedure (see Fig. 1) to discover the attacks that find the best minimally-perturbed adversarial examples with fewer queries to the models. AttackBench ranks the attacks based on a novel metric, called *optimality*, that quantifies how close each attack is to the empirically-optimal solution (estimated by ensembling all the attacks tested). The optimality metric evaluates the effectiveness of gradient-based attacks across

*These authors contributed equally.

various perturbation budgets, avoiding pointwise evaluations at fixed perturbation budgets, and promoting attacks that consistently find smaller perturbations to evade the target model. AttackBench also compares the attacks in their computational efficiency, measuring their runtime within the given maximum computational budget. We then use AttackBench to perform an extensive benchmark analysis that compares 20 attacks (listed in table 3). For each attack, we consider all the implementations available among popular adversarial attack libraries and the original authors' code when available. We empirically test a total of 102 techniques, re-evaluating them in terms of their runtime, success rate and perturbation size, as well as our newly introduced *optimality* metric. While developing our benchmark, we uncovered further insights, including suboptimal implementations and source code errors that prevent some attacks from completing their runs correctly. To foster reproducible results, we open source AttackBench and will provide an online leaderboard, enabling researchers to easily assess the performance of newly-proposed attacks against competing strategies in a unified setting.

We summarize our contributions as follows: (i) we propose AttackBench, a fair benchmark for adversarial attacks that evaluates them on consistent models, data, and computational budgets, and introduce the novel *optimality* metric to rank them based on the quality of the adversarial examples they generate; (ii) we extensively test 102 attacks and we rank them according to our novel metric (Sec.4); and lastly (iii) we highlight 5 programming errors inside the code of some attacks we have considered, and we present inconsistent results of different implementation of the same attack (Sec.4).

2 Gradient-based Attacks

Let us assume, without loss of generality, that the input samples lie in a d-dimensional (bounded) space, *i.e.* $\mathbf{x} \in [0,1]^{\mathsf{d}}$, and that their labels are denoted with $y \in \{1,...,C\}$. Then, the predicted label of a trained model parameterized by $\boldsymbol{\theta}$ can be denoted with $\hat{y} = f(\mathbf{x}, \boldsymbol{\theta})$, while the confidence value (logit) for class c can be denoted with $f_c(\mathbf{x}, \boldsymbol{\theta})$ and the softmax-rescaled logits with $z_c(\mathbf{x}, \boldsymbol{\theta})$. The predicted label can thus be computed also as $\hat{y} = \arg\max_c f_c(\mathbf{x}, \boldsymbol{\theta})$. Under this setting, finding an adversarial example amounts to solving the following multi-objective optimization:

$$\underset{\boldsymbol{\delta}}{\text{minimize}} \qquad (L(\mathbf{x}+\boldsymbol{\delta}, y; \boldsymbol{\theta}), \|\boldsymbol{\delta}\|_p)\ , \tag{1}$$

$$\text{subject to} \qquad \mathbf{x}+\boldsymbol{\delta} \in [0,1]^{\mathsf{d}}\,, \tag{2}$$

where $L(\mathbf{x}+\boldsymbol{\delta}, y; \boldsymbol{\theta})$ is a loss defining the misclassification objective, and $\boldsymbol{\delta}$ is the perturbation optimized to find an adversarial example $\mathbf{x}' = \mathbf{x} + \boldsymbol{\delta}$ within the feasible domain. The loss function $L : \mathbb{R}^{\mathsf{d}} \times \mathbb{R} \to \mathbb{R}$ defines the objective of the attack so that L is large when the input is correctly classified, whereas it is lower when the model predicts a wrong label for $\mathbf{x}$.[1] The majority of existing attacks now leverage the Negative Cross-Entropy (`NCE`) loss, the Difference of Logits (`DL`) (Carlini and Wagner 2017), or the Difference of Logits Ratio (`DLR`) (Croce and Hein 2020b). The second objective in Eq. (1) is expressed as a constraint on the size of the perturbation $||\boldsymbol{\delta}||_p$, formulated through the usage of ℓ_p norms. Typically, ℓ_0, ℓ_1, ℓ_2, and ℓ_∞ norms are used, yielding sparse to increasingly dense adversarial perturbations. The box constraint in Eq. (2) ensures that the sample remains within the input space of the model, *i.e.* $\mathbf{x}+\boldsymbol{\delta} \in [0,1]^{\mathsf{d}}$.

The optimization problem expressed in Eq. (1) presents an inherent tradeoff: minimizing L favors the computation of adversarial examples with large misclassification confidence, but also a large perturbation size, whereas minimizing $\|\boldsymbol{\delta}\|_p$ penalizes larger perturbations at the expense of decreasing the misclassification confidence. We can thus define two main families of attacks, where one aims at finding the inputs that cause the maximum error within a given perturbation budget ε (`FixedBudget`) (Biggio et al. 2013; Madry et al. 2018), and the counterpart that searches for the smallest perturbation needed to achieve misclassification (`MinNorm`) (Szegedy et al. 2014; Brendel et al. 2020).

Limitations of Attack Evaluation Metrics. We discuss here the main metrics that are normally used to evaluate the effectiveness of `FixedBudget` and `MinNorm` attacks. For `FixedBudget` attacks, the Attack Success Rate $\text{ASR}_a(\varepsilon)$ is used to determine the success rate of attack a when the perturbation budget is at most ε (Madry et al. 2018; Croce and Hein 2019, 2021, 2020a). It is defined as:

$$\text{ASR}_a(\varepsilon) = \frac{1}{|\mathcal{D}|} \sum_{(\mathbf{x},y)\in\mathcal{D}} \mathbb{I}(d_{\mathbf{x}} \le \varepsilon)\,, \tag{3}$$

where $d_{\mathbf{x}} = \|\mathbf{x}_{\text{adv}} - \mathbf{x}\|_p$ is the perturbation size of the adversarial examples that successfully mislead the prediction of the target model, and $\mathbb{I}(\cdot)$ is the indicator function, which returns 1 if its argument is true, and 0 otherwise.

Evaluations for `MinNorm` attacks focus on the mean (Carlini and Wagner 2017; Chen et al. 2018) or median (Brendel et al. 2020; Cinà et al. 2024) values of the adversarial perturbation. The median in particular indicates the perturbation budget at which 50% of attacks are successful.

Unfortunately, these metrics suffer from some limitations. In particular, (i) the ASR_a and the median do not allow a direct comparison of the performance of `FixedBudget` and `MinNorm` attacks. Moreover, (ii) these metrics do not capture whether an attack is succeeding or not at larger perturbation budgets, which is instead useful to understand whether the attack is failing due, e.g., to implementation issues or for being ineffective against the specific model (Carlini et al. 2019; Pintor et al. 2022). Finally, (iii) these metrics do not provide any insights on how close or far the results of each attack are from the best possible (though unknown) solution. While newly-proposed attacks are normally compared with a bunch of competing approaches, as we will see, these evaluations still fall short when it comes to assess the effectiveness of each attack with respect to the optimal solution.

3 The AttackBench Framework

We now describe the stages of AttackBench, depicted in Figure 1. In Section 3.1, we detail stages (1-2), which define the benchmark environment setup for executing attacks under

[1] For targeted attacks, one can minimize $L(\mathbf{x}+\boldsymbol{\delta}, y_t; \boldsymbol{\theta})$, being $y_t \neq y$ the label of the target class.

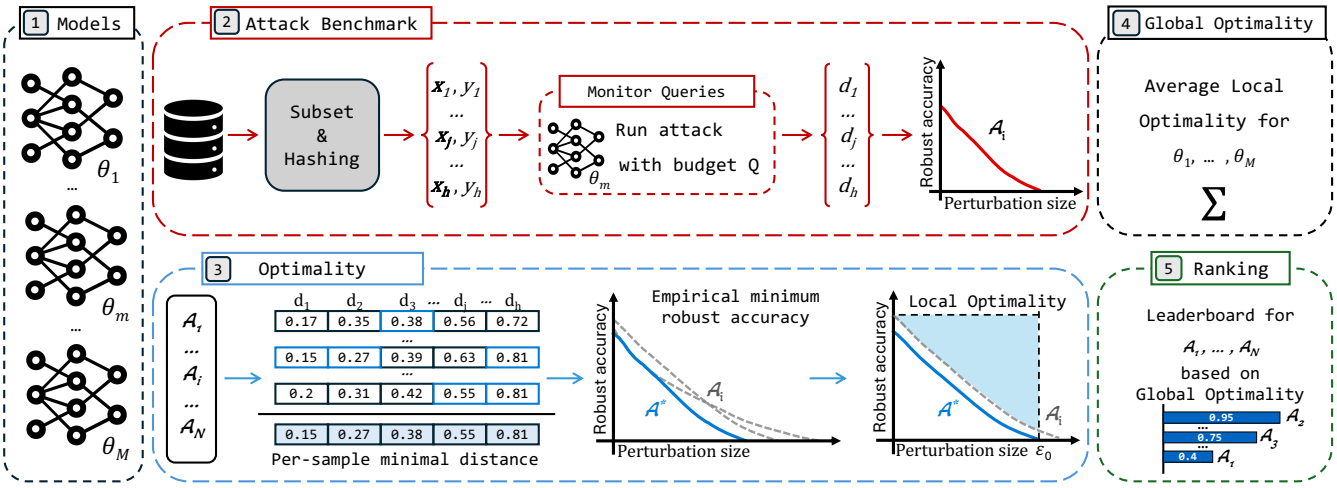

Figure 1: A comprehensive overview of the five stages of AttackBench. Each attack is tested in fair conditions and ranked through the optimality metric. The best attack is the one that produces minimally-perturbed adversarial examples with fewer queries.

Algorithm 1: Attack Benchmarking

Input : a, the attack algorithm; $\boldsymbol{\theta}$, the target model; $\mathcal{D}$, the test dataset; Q, the query budget; and $p \in \{0, 1, 2, \infty\}$, the perturbation model.

Output : Min. distances $d^\star$ and required queries $q^\star$.

1 $d^\star \leftarrow \{\}, \quad q^\star \leftarrow \{\}$
2 $\texttt{B}_{\boldsymbol{\theta}} \leftarrow \texttt{BenchModel}(\boldsymbol{\theta}, Q)$
3 **for** $(\mathbf{x}, y) \in \mathcal{D}$ **do**
4 $\quad \texttt{B}_{\boldsymbol{\theta}}\texttt{.init\_queries()}$
5 $\quad h_x \leftarrow \texttt{hash}(\mathbf{x})$
6 $\quad \mathbf{x}_{\text{adv}} \leftarrow a(\mathbf{x}, y, \texttt{B}_{\boldsymbol{\theta}})$
7 $\quad d^\star\{h_x\} \leftarrow \|\mathbf{x} - \mathbf{x}_{\text{adv}}\|_p$
8 $\quad q^\star\{h_x\} \leftarrow \texttt{B}_{\boldsymbol{\theta}}\texttt{.num\_queries()}$
9 **return** $d^\star, q^\star$

standardized conditions. We then present in Section 3.2 stages (3-4-5), where we aggregate results from stage (2) to compute a novel *optimality* metric, as shown in stage (3) of Figure 1, for ranking gradient-based attacks. Lastly, we explain how to integrate novel attacks into the benchmark and provide implementation details in Appendix B.

3.1 Model Zoo and Attack Benchmarking

Stage (1): Model Zoo. It builds a *zoo* of diverse models, encompassing robust and non-robust models, to evaluate attacks across various scenarios and minimize evaluation biases. The pool can be clearly expanded as novel defenses are released.

Stage (2): Attack Benchmarking. It consists of testing each attack with Algorithm 1, against all models in the AttackBench zoo. For each attack a, target model $\boldsymbol{\theta}$, and dataset $\mathcal{D}$, AttackBench returns the best minimum distances $d^\star$ found by a for each sample, as well as the number of forward and backward queries $q^\star$ utilized. In particular, Algorithm 1 begins by initializing the optimal minimum distances $d^\star$ and $q^\star$. It then wraps the target model $\boldsymbol{\theta}$ within a custom object $\texttt{B}_{\boldsymbol{\theta}}$, which tracks the number of queries—both forward and backward passes—used by the attack to optimize each sample (Line 2). The algorithm then iterates over the input samples $\mathbf{x}$ and their labels y in $\mathcal{D}$. The loop starts by setting the number of queries for the current sample to zero (Line 4), and then hashes the sample (Line 5) to ensure consistent result storage, even if the samples are processed in a different order. The attack is then executed (Line 6), while tracking the number of queries made to the target model. If the query budget Q is exceeded, the attack is halted. Whether the attack exceeds Q or not, AttackBench always returns the best adversarial example $\mathbf{x}_{\text{adv}}$ found during the optimization. This approach improves upon many existing attack implementations, which typically return only the last sample, even though it may not be the optimal one (Pintor et al. 2022). The algorithm then calculates the distance $d^\star$ between the adversarial sample $\mathbf{x}_{\text{adv}}$ and the source sample $\mathbf{x}$ (Line 7), and retrieves the number of queries $q^\star$ used to optimize $\mathbf{x}_{\text{adv}}$ from $\texttt{B}_{\boldsymbol{\theta}}$ (Line 8). Finally, the results ($d^\star$, $q^\star$) for all samples are returned and used to compute the robustness evaluation curve. Remarkably, AttackBench enables the benchmarking of novel attacks without altering their code or adjusting their parameters, avoiding misconfiguration issues. AttackBench operates by wrapping the model and continuously monitoring all queries made by the attack to provide a comprehensive view of the attack's performance and resource usage. Technical details regarding the implementation of stage (2) are reported Appendix B.

3.2 Evaluating and Ranking Attacks

Stage (3): Optimality Score. AttackBench compares gradient-based attacks by inspecting the robustness evaluation curves (Biggio, Fumera, and Roli 2014; Biggio and Roli 2018) they generate against the target models in the *zoo*. These curves illustrate how robust accuracy $\rho_a(\varepsilon)$, i.e., 1 - $\text{ASR}_a(\varepsilon)$, decreases as the perturbation budget ε for attack a increases. Some examples are shown in Figure 2. Note that when ε equals zero, $\rho_a(0) = \rho$ corresponds to the model's *clean accuracy*, meaning the accuracy on unperturbed sam-

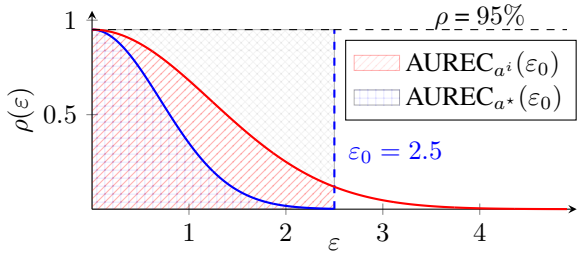

Figure 2: Robustness evaluation curves of a^i and $a^\star$.

ples. More effective attacks typically produce robustness evaluation curves closer to the origin. The Area Under the Robustness Evaluation Curve (AUREC) can be defined as:

$$\text{AUREC}_a(\varepsilon_0) = \int_0^{\varepsilon_0} \rho_a(\varepsilon)\mathrm{d}\varepsilon, \tag{4}$$

where ε_0 denotes the upper bound of the integration interval, which should be set to ∞ or to the smallest ε at which $\rho_a(\varepsilon) = 0$ to compute the complete area. The AUREC can be used to summarize the average attack effectiveness on the whole robustness evaluation curve, as illustrated in Figure 2.

However, AUREC values obtained from different models are not directly comparable, as the models tend to exhibit different *clean accuracy* values. As the outcome of our benchmark's stage (3), we overcome this issue by proposing a novel metric, called *local optimality* and denoted with $\xi^i_{\boldsymbol{\theta}}$. To this end, we first define the best (empirical) attack $a^\star$, by ensembling all the attacks tested in our benchmark. In particular, for each sample, this amounts to setting the best minimum distance $d^\star$ of $a^\star$ as the smallest distance found among *all* the considered attacks (see Figure 1). As we will see in our experiments, this provides the best possible empirical estimate of the optimal solution for each sample.[2] We then set ε_0 as the smallest value at which $\rho^\star_a(\varepsilon) = 0$, and compute AUREC$_{a^\star}(\varepsilon_0)$ from the robustness evaluation curve of $a^\star$.

Local optimality (LO). The *LO* metric, denoted as $\xi^i_{\boldsymbol{\theta}}$, computes the difference between the area under the curve of a^i and $a^\star$ against the target $\boldsymbol{\theta}$ to evaluate how much the first is suboptimal with respect to the second. Essentially, *LO* indicates the loss in performance of a^i compared to $a^\star$. In Figure 2, we illustrate an example of the *LO* measure when comparing the robustness evaluation curve of a^i (red curve) with that of the best attack, $a^\star$ (blue curve). The *LO* metric quantifies the area of the gray region above the red curve in Figure 2, inversely proportional to the area between the two curves. When the area between these two curves is non-zero, it indicates that a^i is less optimal than $a^\star$. The adversarial examples generated by a^i thus are generally sub-optimal, meaning they succeed only with larger perturbations compared to the empirical lower bound found by $a^\star$. Formally:

$$\xi^i_{\boldsymbol{\theta}} = \frac{\rho(\varepsilon_0)\cdot\varepsilon_0 - \text{AUREC}_{a^i}(\varepsilon_0)}{\rho(\varepsilon_0)\cdot\varepsilon_0 - \text{AUREC}_{a^\star}(\varepsilon_0)}, \tag{5}$$

[2]Let us remark that we are ensembling over 100 attacks in AttackBench, whereas previous evaluations normally consider less than tens of competing approaches without even ensembling them.

where the box defined by $\rho\cdot\varepsilon_0$ serves to normalize the *LO* measure with respect to the clean accuracy ρ and robustness ε_0 of the target model $\boldsymbol{\theta}$. In more detail, when a target model exhibits high robustness (i.e., large ε_0) or accuracy (i.e., large ρ), evading the target model becomes inherently more challenging, leading to less severe penalties for the optimality of the attack. Conversely, when the target model is more vulnerable or less accurate, the curve disparity becomes more relevant, and the impact on the optimality of the attacks is more pronounced. Furthermore, this term ensures that the *LO* measure is bounded in $[0,1]$ with $\xi^i_{\boldsymbol{\theta}} = 1$ when a^i performs exactly as $a^\star$, thus finds always the smallest perturbation to evade the target model for each sample. By contrast, when $\xi^i_{\boldsymbol{\theta}} = 0$, the attack fails to find a successful perturbation with $\varepsilon \leq \varepsilon_0$, thus resulting in a curve that matches the box and therefore AUREC$_{a^i}(\varepsilon_0) = \rho\cdot\varepsilon_0$. The *LO* measure thus offers a clear scale to interpret and compare the performance of gradient-based attacks across different models. Lastly, stage (3) of AttackBench uses a broad range of attacks to construct $a^\star$ and assess, with the *LO* measure, how far an attack is from an ideal top performer across multiple perturbation budgets.

Stage (4): Global Optimality. The *LO* measure evaluates the effectiveness of an attack against a single target model, $\boldsymbol{\theta}_m$. However, ranking attacks with respect to a single model increases the risk of overfitting the benchmark. An attack might be specifically tailored for a particular model $\boldsymbol{\theta}_m$, but have poor generalization and lower performance across different models. To this end, AttackBench prevents potential bias by evaluating the attacks against a diverse set of robust models and aggregating the results to derive a global score, namely *global optimality*, as the output of step (4).

Global Optimality (GO). The *GO* measure quantifies the average *LO* of an attack, denoted as a^i, concerning a set of target models $\mathcal{M} = \{\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_M\}$. We define the *GO* as:

$$\xi^i = \frac{1}{|\mathcal{M}|}\sum_{\boldsymbol{\theta}_m\in\mathcal{M}} \xi^i_{\boldsymbol{\theta}_m}. \tag{6}$$

The *GO* score for an attack corresponds to the average *LO* measure obtained across a set $\mathcal{M}$ of distinct models. Bounded in $[0,1]$ as well, the *GO* score for an attack a^i equals 1 only when a^i consistently identifies the minimal adversarial perturbations for all input samples across every model in $\mathcal{M}$, meaning a^i performs as well as $a^\star$ in every case (i.e., $a^i = a^\star$).

Stage (5): Ranking. Lastly, AttackBench ranks attacks by their *GO*, grouping them based on the ℓ_p norm threat model they consider and the target model. Remarkably, the AttackBench *optimality* ranking can be continuously and efficiently updated as new attacks emerge, without the need to re-run previous ones. When a new attack is uploaded, AttackBench updates the best minimum distances $d^\star$ of $a^\star$ and refreshes the leaderboard for all attacks by recalculating the *LO* score employing the stored robustness evaluation curves (additional details in Appendix B.4).

4 Experiments

We now execute AttackBench to rate the adversarial attacks.

Algorithm 2: Computing Local Optimality

Input : $\{a^1, \dots, a^N\}$, list of attacks for benchmark; $\mathcal{D}$, the validation dataset; $\boldsymbol{\theta}$, the target model; Q, query budget; p, threat model distance.
Output : Local optimality measure $(\xi^1_{\boldsymbol{\theta}}, \dots, \xi^n_{\boldsymbol{\theta}})$.

for $a^i \in \{a^1, \dots, a^N\}$ **do**
  $d_i, q_i \leftarrow \texttt{TestAttack}(a^i, \mathcal{D}, \boldsymbol{\theta}, Q, p)$
$\rho^*_{\boldsymbol{\theta}}(\epsilon) = \texttt{sample-wise-min}\{d_1, \dots, d_N\}$
for $a^i \in \{a^1, \dots, a^n\}$ **do**
  Get $\xi^i_{\boldsymbol{\theta}}$ for attack a^i following eq. (5)
return $(\xi^1_{\boldsymbol{\theta}}, \dots, \xi^n_{\boldsymbol{\theta}})$, $(q_1, \dots, q_n)$

4.1 Experimental Setup

Dataset. We consider two popular datasets: CIFAR-10 (Krizhevsky 2009) and ImageNet (Deng et al. 2009). We evaluate the performance of adversarial attacks on the entire CIFAR-10 test set, and on a random subset of 5 000 samples from the ImageNet validation set. We use a batch size of 128 for CIFAR-10 and 32 for ImageNet.

Models. AttackBench models *zoo* utilizes a selection of both baseline and robust models to evaluate the effectiveness and adaptability of attacks against various model architectures and defense mechanisms. For CIFAR-10, we include a baseline undefended WideResNet-28-10 (denoted as C1) from Robustbench (Croce et al. 2021) and four robust models implementing the following defenses: a certified defense (Zhang et al. 2020) (C2); adversarial training for generalization against unseen attacks (Stutz, Hein, and Schiele 2019) (C3); gradient obfuscation defense (Xiao, Zhong, and Zheng 2020) (C4); and adversarial training with data augmentation (Wang et al. 2023) (C5). For ImageNet, we select four models from Robustbench (Croce et al. 2021): a pre-trained undefended ResNet-50, and three adversarially-trained models by Wong, Rice, and Kolter (2020) (I2), Salman et al. (2020) (I3), and Debenedetti, Sehwag, and Mittal (2023) (I4).

Adversarial Libraries. We integrate six publicly available adversarial attack libraries in AttackBench: FoolBox (FB) (Rauber et al. 2020), CleverHans (CH) (Papernot et al. 2018), AdvLib (AL) (Rony and Ben Ayed 2024), ART (Art) (Nicolae et al. 2018), TorchAttacks (TA) (Kim 2020), and DeepRobust (DR) (Li et al. 2020). We also include authors' original implementations of their attacks to verify implementation inconsistencies (or improvements) among the different versions. We denote them denoted with O.

Attack Settings. We focus on the well-studied ℓ_p perturbation models with $p \in \{0, 1, 2, \infty\}$. We consider all the available implementations of the attacks listed in Table 3. We only run the attacks in their untargeted objective. [3]

Hyperparameters. For each considered attack implementation, we employed the default hyperparameters. We set the maximum number of forward and backward propagations Q to 2 000. For an attack that does a single forward prediction and backward gradient computation per optimization step, this corresponds to the common "1 000 steps budget" found in several works (Brendel, Rauber, and Bethge 2018; Rony et al. 2019; Pintor et al. 2021; Rony et al. 2021), sufficient for algorithm convergence (Pintor et al. 2022). Exceptionally, most implementations of the CW attack use a default number of steps equal to 10^4 for multiple runs to find c, so we modify it to perform the search of the penalty weight and the attack iterations within the $2,000$ propagations. Furthermore, as done in (Rony et al. 2021; Cinà et al. 2024), for `FixedBudget` attacks we leverage a line-search strategy to find the smallest budget $\varepsilon^\star$ for which the attack can successfully find an adversarial perturbation. Further details on the line search are reported in Appendix B.5.

Evaluation Metrics. For each attack, we compute the Attack Success Rate (ASR) as the percentage of samples in $\mathcal{D}$ that the attack converts into adversarial examples at $\varepsilon = \infty$ which, if different from zero, indicates that the attack is applied incorrectly (Carlini et al. 2019), and the *local* and *global optimality* scores. Additionally, we track the computational effort of each attack expressed in terms of execution time and the number of forward and backward propagations required. The execution time is measured on a shared compute cluster equipped with NVIDIA V100 SXM2 GPU (16GB memory).

4.2 Experimental Results

We present the benchmark results of AttackBench on a total of **815** experiments, encompassing the 102 state-of-the-art gradient-based attacks implementations listed in Table 3 (see Appendix A). For the 5 CIFAR-10 models, we execute $6, 21, 42$, and 33 implementations for the ℓ_0, ℓ_1, ℓ_2, and ℓ_∞ perturbation models respectively, obtaining a total of 510 experiments. For the 4 ImageNet models, given the high data dimensionality, we reduced the load by excluding some implementations that were suboptimal in the CIFAR-10 experiments. Therefore, we execute $6, 10, 9$, and 10 attack implementations for ℓ_0, ℓ_1, ℓ_2, and ℓ_∞, obtaining 140 experiments.

Attack Optimality. Section 4.2 shows the results obtained for the top-5 ranked attacks in each ℓ_p norm, according to our *GO* metric. Complementary, we report the 5 worst-performing attacks in Section 4.2. We defer to Appendix C additional results, including remaining attacks and singular model evaluations. We also visually depict examples of the robustness evaluation curves obtained with the top-ranked attacks in Figure 3 computed against CIFAR-10 model C3 (Stutz, Hein, and Schiele 2019). We observe that the top-ranked attacks for the ℓ_0 and ℓ_2 norms are σ-zero (Cinà et al. 2024) and DDN (Rony et al. 2019) for both CIFAR-10 and ImageNet. In their respective ℓ_p norms, these attacks achieve *GO* scores very close to 1, indicating they are nearly as effective as the empirical best attack $a^\star$ (blue curve in Figure 2, black curve in Figure 3). Regarding the ℓ_1 and ℓ_∞ threat models, we observe the constant presence of APGD-ℓ_1 (Croce and Hein 2021), APGD$_t$ (Croce and Hein 2020b), and PDPGD (Matyasko and Chau 2021) in the top 5 positions in both CIFAR-10 and ImageNet experiments. PDPGD is, for example, the top-ranked attack in ℓ_1 for CIFAR-10 and second on ImageNet by a small margin. In both cases, APGD competes with PDPGD. In ℓ_∞, we observe that APGD$_t$ and APGD demonstrate similar results, both contending for the lead. We highlight that APGD, despite being

[3] APGD$_t$ runs APGD with a targeted objective on multiple classes, retaining the best result as an untargeted solution.

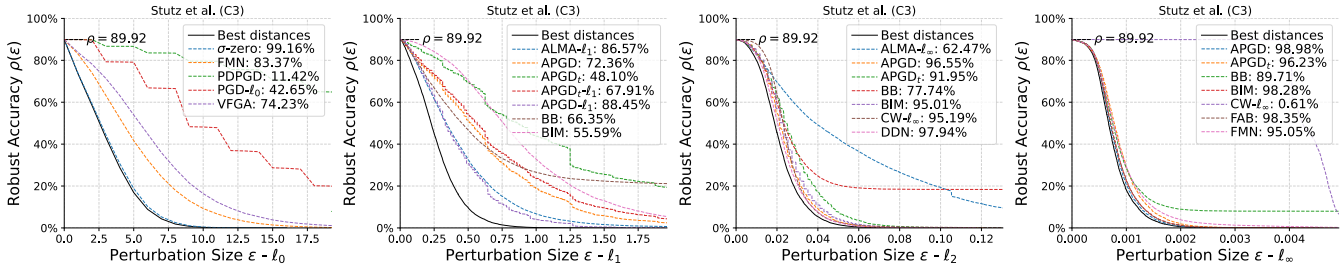

Figure 3: Robustness evaluation curves for the 7 best ℓ_0, ℓ_1, ℓ_2, and ℓ_∞-norm attacks against C3 (Stutz, Hein, and Schiele 2019).

conceived as a `FixedBudget` attack, offers outstanding results also when configured as a `MinNorm` attack. This suggests that combining APGD with the search strategy presented in Section 4.1 is a viable strategy to efficiently find a small perturbation, even compared to `MinNorm` attacks. Note finally that none of the tested attacks reaches 100% of *GO*, suggesting that combining attacks, as done by Croce et al. (2021), can lead to improved robustness evaluations.

Fast Attacks. The top-ranked attacks in Section 4.2 do not always offer the best effectiveness/efficiency tradeoff. For instance, in the ℓ_1 norm, APGD achieves only a slightly better *GO* score than PDPGD on ImageNet but is three times slower due to its original implementation and the integration of the ϵ search strategy. In the ℓ_0 norm, the VFGA (Hajri et al. 2020) attack stands out as the fastest, requiring significantly fewer queries due to its early stopping mechanism. However, its ASR and *GO* largely drop for this attack when scaling to the ImageNet dataset, diminishing its reliability.

Best and Worst Performing Implementations. From Section 4.2 we observe that high-ranked attacks are frequently implemented in AdvLib and Foolbox libraries or are directly taken from the original repositories of the authors. Conversely, implementations from other libraries, such as the Art, Cleverhans, and Deeprobust, do not consistently appear among the top-performing attacks. In some cases, attacks integrated into these libraries even exhibit a decrease in performance. For example, while APGD implemented with AdvLib (as well as its original implementation) ranks among the best-performing attacks (see Section 4.2) its implementation in the Art library ranks among the worst (see Section 4.2). Specifically, for the CIFAR-10 dataset under the ℓ_1 norm threat model, *GO* optimality drops from 90.9% (original implementation) to 26% (Art). Upon inspecting the code, we found a key difference in the parameter controlling the number of random initializations for the attack: it is set to 5 in Art and 1 in both the original and AdvLib implementations. Consequently, the restart mechanism of APGD consumes more queries within the same perturbation budget ε, leading to early termination of the attack without exploring smaller budgets. Another crucial difference is observed for FAB (GitHub 2024b) and CW (GitHub 2024a) because they calculate the difference of the gradients of the two logits separately, thus requiring two backward passes rather than one. A final notable divergence in implementation is observed in the ImageNet results for the APGD attack between the original repository and the AdvLib library. In the ℓ_2 and ℓ_∞ norms, AdvLib's APGD achieves *GO* scores of 95.9% and 97%, respectively, compared to 36.2% and 39.1% for the original APGD, representing substantial gaps of 59.7% and 57.9%. We also observed that while the original implementations used the `NCE` loss, AdvLib defaults to the `DLR` loss.

Ingredients for Optimal Attacks. We here identify key factors contributing to a high *GO* score in Section 4.2. First, nearly all top-ranked attacks, including σ-zero, FMN, PDPGD, PGD-ℓ_0 (Croce and Hein 2019), APGD, APGD$_t$, BIM (Kurakin, Goodfellow, and Bengio 2016), and DDN, use normalization or linear projections on the gradient. These methods decouple the gradient's original size from the step size used in the updates (Rony et al. 2019; Pintor et al. 2021). Furthermore, top-ranked attacks (e.g., σ-zero, FMN, PDPGD) are usually equipped with dynamic or adaptive step size schedulers, such as cosine annealing or reduce on plateau. None of the attacks securing the top rank in Section 4.2 adopts a fixed step size, confirming that dynamically reducing the step size across iterations and enhancing convergence stability contributes to achieving better optima. Lastly, among the best-performing attacks, optimizers like Adam or momentum variants of gradient descent are only occasionally used.

Implementation Pitfalls. Through AttackBench, we found many coding errors across diverse libraries: (i) the original BB implementation crashes if not all samples are misclassified after initialization; (ii) TorchAttacks DeepFool fails during batch processing if a sample has a label equal to the number of classes, resulting in an index out-of-bounds error; (iii) Foolbox FMN ℓ_0 crashes with out-of-bounds errors during the projection step. Lastly, per-sample ε evaluations in `FixedBudget` attacks are supported only in AdvLib.

5 Related Work

Although many gradient-based attacks for optimizing adversarial examples have been proposed, their performance is often evaluated independently in each study, increasing the risk of evaluation bias and hindering reproducibility. These evaluations do not ensure that attack comparisons are conducted under the same fair experimental settings (e.g., consistent query budgets or data subsets). Moreover, they do not usually compare `FixedBudget` or `MinNorm` under the same evaluation metric, and they do not ensure consistency with the original implementation of the attack, which can induce to misleading conclusions (Carlini 2019).

Researchers have also proposed some benchmarks to stan-

	ℓ_p	Attack	Library	ASR	GO	#F	#B	t(s)
CIFAR-10	ℓ_0	σ-zero	O	100	98.4	999	999	292.2
		FMN	O, *AL*	98.7	85.3	1000	1000	278.8
		VFGA	AL	94.4	80.2	388	18	106.2
		PGD-ℓ_0	O	100	66.7	919	901	545.0
		PDPGD	AL	99.5	39.3	913	913	280.4
		PDPGD	AL	99.8	93.2	995	995	279.6
	ℓ_1	APGD-ℓ_1	*O*, AL	100	90.9	775	755	892.4
		FMN	*O*, AL, FB	97.9	90.4	1000	1000	276.0
		APGD$_t$	*O*, AL	100	85.4	577	536	860.6
		EAD	FB	100	70	923	923	276.7
		DDN	*AL*, FB	100	92.9	998	998	278.0
	ℓ_2	APGD	*O*, AL	100	92.9	775	755	709.2
		APGD$_t$	*O*, AL	100	92.2	522	482	641.8
		PDGD	AL	99	91.7	994	994	279.6
		FMN	O, *AL*	99.5	90.8	998	998	275.3
		APGD$_t$	*O*, AL	100	97.6	629	584	626.1
		APGD	*O*, AL	100	97.5	775	755	711.5
	ℓ_∞	BIM	FB	99.9	94.6	999	989	692.3
		PGD	AL	100	93.2	1000	990	281.8
		PDPGD	AL	99.8	90.8	992	992	284.6
ImageNet	ℓ_0	σ-zero	O	100	99.8	999	999	345.2
		FMN	O, *AL*	100	87.5	1000	1000	358.8
		VFGA	AL	63.6	71.2	928	44	76.4
		PDPGD	AL	100	53.6	983	983	345.1
		PGD-ℓ_0	O	100	39.2	913	893	1680.1
		APGD	*O*, AL	100	98	693	673	1085.9
	ℓ_1	PDPGD	AL	100	97.3	999	999	339.2
		APGD$_t$	*O*, AL	100	94.8	579	538	990.1
		ALMA	AL	100	92.5	973	973	359.3
		EAD	FB	100	90.7	403	206	77.5
		DDN	AL, *FB*	100	98.6	1000	1000	336.0
	ℓ_2	APGD$_t$	AL	99.9	98.1	670	650	325.1
		BIM	FB	100	97.3	1000	990	1252.4
		FMN	O, *AL*	99.6	97.1	1000	1000	362.8
		APGD	O, *AL*	100	95.9	1000	990	333.5
		APGD$_t$	AL	99.8	99.2	718	698	358.7
		PDPGD	AL	100	98.2	987	987	409.1
	ℓ_∞	FMN	*O*, AL, FB	100	98.2	1000	1000	335.2
		BIM	FB	100	98	1000	990	1293.7
		APGD	AL	100	97	1000	990	334.0

Table 1: Top-performing attacks. For each attack, we list the library implementations that yield similar results, highlighting the best one in italics and report its corresponding statics. We include the number of forward (**#F**) and backward (**#B**) passes, as well as the total runtime (**t(s)**) for each attack.

dardize evaluations of models' robustness. Ling et al. (2019) propose DeepSec, which tests 16 state-of-the-art adversarial attacks against 13 robust models computing 12 different metrics defined by the authors to evaluate the attacks' utility. However, DeepSec considers only a fixed arbitrary perturbation budget, and it does not provide a metric that can be used to rank the attacks' effectiveness. Additionally, DeepSec has faced significant criticism for its implementation flaws (Carlini 2019). Frameworks like (Croce et al. 2021; Dong et al. 2020; Guo et al. 2023) focus on evaluating model robustness, not the attacks themselves, often overlooking factors like computational effort. Unlike (Croce et al. 2021; Guo et al. 2023), which evaluate only at a single, arbitrary perturbation budget, Dong et al. (2020) use robustness evaluation curves to compare model robustness, but this approach lacks an objective metric, relying instead on visual inspection of the curves. Additionally, these curves are not directly comparable across models with different *clean accuracy*, making it difficult to identify the most effective attacks. In summary, these robustness evaluation benchmarks propose metrics that are not combinable across models to reflect the global performance of attacks, and they often ignore computational efficiency. AttackBench addresses these limitations by providing an evaluation framework that ranks adversarial attacks based on a single metric that accounts for their effectiveness and efficiency on a consistent experimental setup covering multiple robust models and attack implementations.

ℓ_p	Attack	Library	ASR	GO	#F	#B	t(s)
ℓ_1	PGD	FB	100	55.6	1000	990	715.0
	EAD	Art	85.2	53.3	334	1665	295.7
	FGM	Art, *FB*	97.7	28	40	20	30.3
	APGD	Art	98.8	25.6	822	354	456.9
	BB	FB	38	38	623	36	119.4
ℓ_2	DeepFool	FB	98.6	40.6	256	255	21.2
	FGM	Art, *CH*, DR, FB	97.6	37.9	41	20	28.1
	DeepFool	Art	84.9	32.3	269	1341	317.8
	BB	FB	38.3	30.9	624	36	112.1
	BIM	Art	95.7	22.6	808	782	322.2
ℓ_∞	APGD	Art	94.5	77.5	1037	504	390.0
	FGSM	*TA, FB, DR, CH, Art*	97.6	62.9	40	20	7.9
	CW	*Art*, AdvLib	86.2	62.5	1321	640	2314.4
	DeepFool	FB	98.3	46.8	129	128	64.1
	BB	FB	42.9	32	806	135	139.0

Table 2: Worst performing attacks on CIFAR-10. For each attack, we list the library implementations that yield similar results, highlighting the worst one in italics with its statistics. We include the number of forward (**#F**) and backward (**#B**) passes, as well as the total runtime (**t(s)**) for each attack.

6 Conclusion and Future Work

In this work, we propose AttackBench, a comprehensive benchmark to evaluate gradient-based attacks, which stands on top of the proposed *optimality* metric. This novel measure is the first that allows ranking attack algorithms according to their effectiveness across their entire robustness evaluation curve. Employing AttackBench, we enable researchers to evaluate attacks effectiveness in fair conditions that ensure both reproducibility and impartial ranking. Thanks to AttackBench, we can spotlight attacks that excel in most of the perturbation models while also alerting the presence of critical errors of libraries that prevent some strategies from properly running. AttackBench, currently restricted to evaluating gradient-based attacks on image models, is a versatile methodology with the potential to be extended in future work to black-box attacks and other domains (e.g., malware).

Ethical Statement

We assert that there are no identifiable ethical considerations or foreseeable negative societal consequences that warrant specific attention within the confines of this study. Rather this study will help improve the development of more reliable tools to test the adversarial robustness properties of DNNs.

Acknowledgments

This work has been partially supported by the EU—NGEU National Sustainable Mobility Center (CN00000023), Italian Ministry of University and Research Decree n. 1033—17/06/2022 (Spoke 10); the project Sec4AI4Sec, under the EU's Horizon Europe Research and Innovation Programme (grant agreement no. 101120393); the project ELSA, under the EU's Horizon Europe Research and Innovation Programme (grant agreement no. 101070617); and projects SERICS (PE00000014) and FAIR (PE0000013) under the MUR NRRP funded by the EU—NGEU.

References

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

Biggio, B.; Fumera, G.; and Roli, F. 2014. Security Evaluation of Pattern Classifiers Under Attack. *IEEE Transactions on Knowledge and Data Engineering*, 26(4): 984–996.

Biggio, B.; and Roli, F. 2018. Wild Patterns: Ten Years After the Rise of Adversarial Machine Learning. *Pattern Recognition*, 84: 317–331.

Brendel, W.; Rauber, J.; and Bethge, M. 2018. Decision-Based Adversarial Attacks: Reliable Attacks Against Black-Box Machine Learning Models. In *International Conference on Learning Representations*.

Brendel, W.; Rauber, J.; Kümmerer, M.; Ustyuzhaninov, I.; and Bethge, M. 2020. Accurate, reliable and fast robustness evaluation. In *Thirty-third Conference on Neural Information Processing Systems (NeurIPS 2019)*, 12817–12827. Curran.

Carlini, N. 2019. A critique of the DeepSec Platform for Security Analysis of Deep Learning Models. arXiv:1905.07112.

Carlini, N.; Athalye, A.; Papernot, N.; Brendel, W.; Rauber, J.; Tsipras, D.; Goodfellow, I.; Madry, A.; and Kurakin, A. 2019. On Evaluating Adversarial Robustness. arXiv:1902.06705.

Carlini, N.; Jagielski, M.; Choquette-Choo, C. A.; Paleka, D.; Pearce, W.; Anderson, H.; Terzis, A.; Thomas, K.; and Tramèr, F. 2023. Poisoning web-scale training datasets is practical. *arXiv*.

Carlini, N.; and Wagner, D. A. 2017. Towards Evaluating the Robustness of Neural Networks. In *IEEE Symposium on Security and Privacy*, 39–57. IEEE Computer Society.

Chen, P.-Y.; Sharma, Y.; Zhang, H.; Yi, J.; and Hsieh, C.-J. 2018. Ead: elastic-net attacks to deep neural networks via adversarial examples. In *Thirty-second AAAI conference on artificial intelligence*.

Cinà, A. E.; Villani, F.; Pintor, M.; Schönherr, L.; Biggio, B.; and Pelillo, M. 2024. σ-zero: Gradient-based Optimization of ℓ_0-norm Adversarial Examples. arXiv:2402.01879.

Croce, F.; Andriushchenko, M.; Sehwag, V.; Debenedetti, E.; Flammarion, N.; Chiang, M.; Mittal, P.; and Hein, M. 2021. RobustBench: a standardized adversarial robustness benchmark. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021*.

Croce, F.; and Hein, M. 2019. Sparse and Imperceivable Adversarial Attacks. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 4723–4731.

Croce, F.; and Hein, M. 2020a. Minimally distorted adversarial examples with a fast adaptive boundary attack. In *International Conference on Machine Learning*, 2196–2205. PMLR.

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

Croce, F.; and Hein, M. 2021. Mind the box: l1-APGD for sparse adversarial attacks on image classifiers. In *International Conference on Machine Learning*.

Dang, Q. H. 2012. Secure Hash Standard (SHS).

Debenedetti, E.; Sehwag, V.; and Mittal, P. 2023. A Light Recipe to Train Robust Vision Transformers. In *First IEEE Conference on Secure and Trustworthy Machine Learning*.

Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; and Fei-Fei, L. 2009. Imagenet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Dong, Y.; Fu, Q.-A.; Yang, X.; Pang, T.; Su, H.; Xiao, Z.; and Zhu, J. 2020. Benchmarking Adversarial Robustness on Image Classification. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 318–328.

GitHub. 2024a. C&W Attack. [Accessed 01-08-2024].

GitHub. 2024b. FAB Attack GitHub. [Accessed 01-08-2024].

Goodfellow, I. J.; Shlens, J.; and Szegedy, C. 2015. Explaining and Harnessing Adversarial Examples. In *International Conference on Learning Representations*.

Guo, J.; Bao, W.; Wang, J.; Ma, Y.; Gao, X.; Xiao, G.; Liu, A.; Dong, J.; Liu, X.; and Wu, W. 2023. A comprehensive evaluation framework for deep model robustness. *Pattern Recognition*, 137: 109308.

Hajri, H.; Césaire, M.; Combey, T.; Lamprier, S.; and Gallinari, P. 2020. Stochastic sparse adversarial attacks. *2021 IEEE 33rd International Conference on Tools with Artificial Intelligence (ICTAI)*, 1247–1254.

Kim, H. 2020. Torchattacks : A Pytorch Repository for Adversarial Attacks. *ArXiv*, abs/2010.01950.

Krizhevsky, A. 2009. Learning Multiple Layers of Features from Tiny Images.

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

Li, Y.; Jin, W.; Xu, H.; and Tang, J. 2020. Deeprobust: A pytorch library for adversarial attacks and defenses. *arXiv preprint arXiv:2005.06149*.

Ling, X.; Ji, S.; Zou, J.; Wang, J.; Wu, C.; Li, B.; and Wang, T. 2019. DEEPSEC: A Uniform Platform for Security Analysis of Deep Learning Model. In *IEEE Symposium on Security and Privacy (SP)*, 673–690.

Madry, A.; Makelov, A.; Schmidt, L.; Tsipras, D.; and Vladu, A. 2018. Towards Deep Learning Models Resistant to Adversarial Attacks. In *ICLR*.

Matyasko, A.; and Chau, L.-P. 2021. PDPGD: Primal-Dual Proximal Gradient Descent Adversarial Attack. *ArXiv*, abs/2106.01538.

Modas, A.; Moosavi-Dezfooli, S.-M.; and Frossard, P. 2018. SparseFool: A Few Pixels Make a Big Difference. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 9079–9088.

Moosavi-Dezfooli, S.-M.; Fawzi, A.; and Frossard, P. 2016. Deepfool: a simple and accurate method to fool deep neural networks. In *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2574–2582.

Nicolae, M.-I.; Sinn, M.; Tran, M.-N.; Buesser, B.; Rawat, A.; Wistuba, M.; Zantedeschi, V.; Baracaldo, N.; Chen, B.; Ludwig, H.; Molloy, I.; and Edwards, B. 2018. Adversarial Robustness Toolbox v1.0.0. *arXiv*.

Papernot, N.; Faghri, F.; Carlini, N.; Goodfellow, I.; Feinman, R.; Kurakin, A.; Xie, C.; Sharma, Y.; Brown, T.; Roy, A.; Matyasko, A.; Behzadan, V.; Hambardzumyan, K.; Zhang, Z.; Juang, Y.-L.; Li, Z.; Sheatsley, R.; Garg, A.; Uesato, J.; Gierke, W.; Dong, Y.; Berthelot, D.; Hendricks, P.; Rauber, J.; and Long, R. 2018. Technical Report on the CleverHans v2.1.0 Adversarial Examples Library. *arXiv preprint arXiv:1610.00768*.

Papernot, N.; McDaniel, P.; Jha, S.; Fredrikson, M.; Celik, Z. B.; and Swami, A. 2016. The Limitations of Deep Learning in Adversarial Settings. In *Proc. 1st IEEE European Symposium on Security and Privacy*, 372–387. IEEE.

Pintor, M.; Demetrio, L.; Sotgiu, A.; Demontis, A.; Carlini, N.; Biggio, B.; and Roli, F. 2022. Indicators of attack failure: Debugging and improving optimization of adversarial examples. *Advances in Neural Information Processing Systems*, 35: 23063–23076.

Pintor, M.; Roli, F.; Brendel, W.; and Biggio, B. 2021. Fast Minimum-norm Adversarial Attacks through Adaptive Norm Constraints. In *Thirty-fifth Conference on Neural Information Processing Systems (NeurIPS 2021)*.

Rauber, J.; Zimmermann, R.; Bethge, M.; and Brendel, W. 2020. 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.

Rony, J.; and Ben Ayed, I. 2024. Adversarial Library.

Rony, J.; Granger, E.; Pedersoli, M.; and Ben Ayed, I. 2021. Augmented Lagrangian Adversarial Attacks. In *International Conference on Computer Vision*.

Rony, J.; Hafemann, L. G.; Oliveira, L. S.; Ben Ayed, I.; Sabourin, R.; and Granger, E. 2019. Decoupling direction and norm for efficient gradient-based l2 adversarial attacks and defenses. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 4322–4330.

Salman, H.; Ilyas, A.; Engstrom, L.; Kapoor, A.; and Madry, A. 2020. Do Adversarially Robust ImageNet Models Transfer Better? In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS*.

Stutz, D.; Hein, M.; and Schiele, B. 2019. Confidence-Calibrated Adversarial Training: Generalizing to Unseen Attacks. In *International Conference on Machine Learning*.

Szegedy, C.; Zaremba, W.; Sutskever, I.; Bruna, J.; Erhan, D.; Goodfellow, I.; and Fergus, R. 2014. Intriguing properties of neural networks. In *International Conference on Learning Representations*.

Wang, Z.; Pang, T.; Du, C.; Lin, M.; Liu, W.; and Yan, S. 2023. Better Diffusion Models Further Improve Adversarial Training. In *International Conference on Machine Learning, ICML*, volume 202 of *Proceedings of Machine Learning Research*, 36246–36263. PMLR.

Wong, E.; Rice, L.; and Kolter, J. Z. 2020. Fast is better than free: Revisiting adversarial training. In *8th International Conference on Learning Representations, ICLR*. OpenReview.net.

Xiao, C.; Zhong, P.; and Zheng, C. 2020. Enhancing Adversarial Defense by k-Winners-Take-All. In *International Conference on Learning Representations, ICLR*. OpenReview.net.

Yao, Z.; Gholami, A.; Xu, P.; Keutzer, K.; and Mahoney, M. W. 2018. Trust Region Based Adversarial Attack on Neural Networks. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 11342–11351.

Zhang, H.; Chen, H.; Xiao, C.; Gowal, S.; Stanforth, R.; Li, B.; Boning, D.; and Hsieh, C.-J. 2020. Towards Stable and Efficient Training of Verifiably Robust Neural Networks. In *International Conference on Learning Representations*.