RobustBench: a standardized adversarial robustness benchmark

Francesco Croce*  Maksym Andriushchenko*  Vikash Sehwag*  Edoardo Debenedetti*
Univ. of Tübingen  EPFL  Princeton Univ.  EPFL
Nicolas Flammarion  Mung Chiang  Prateek Mittal  Matthias Hein
EPFL  Purdue Univ.  Princeton Univ.  Univ. of Tübingen

Abstract

As a research community, we are still lacking a systematic understanding of the progress on adversarial robustness which often makes it hard to identify the most promising ideas in training robust models. A key challenge in benchmarking robustness is that its evaluation is often error-prone leading to overestimation of the true robustness of models. While adaptive attacks designed for a particular defense are a potential solution, they have to be highly customized for particular models, which makes it difficult to compare different methods. Our goal is to instead establish a standardized benchmark of adversarial robustness, which as accurately as possible reflects the robustness of the considered models within a reasonable computational budget. To evaluate robustness of models for our benchmark, we consider AutoAttack, an ensemble of white- and black-box attacks which was recently shown in a large-scale study to improve almost all robustness evaluations compared to the original publications. We also impose some restrictions on the admitted models to rule out defenses that only make gradient-based attacks ineffective without improving actual robustness. Our leaderboard, hosted at http://robustbench.github.io/, contains evaluations of 90+ models and aims at reflecting the current state of the art on a set of well-defined tasks in $\ell_\infty$- and $\ell_2$-threat models and on common corruptions, with possible extensions in the future. Additionally, we open-source the library http://github.com/RobustBench/robustbench that provides unified access to 60+ robust models to facilitate their downstream applications. Finally, based on the collected models, we analyze the impact of robustness on the performance on distribution shifts, calibration, out-of-distribution detection, fairness, privacy leakage, smoothness, and transferability.

1 Introduction

Since the finding that state-of-the-art deep learning models are vulnerable to small input perturbations called adversarial examples [121], achieving adversarially robust models has become one of the most studied topics in the machine learning community. The main difficulty of robustness evaluation is that it is a computationally hard problem even for simple $\ell_p$-bounded perturbations [64] and exact approaches [124] do not scale to large enough models. There are already more than 3000 papers on this topic [1], but it is often unclear which defenses against adversarial examples indeed improve robustness and which only make the typically used attacks overestimate the actual robustness. There is an important line of work on recommendations for how to perform adaptive attacks that are selected specifically for a particular defense [5, 16, 127] which have in turn shown that several seemingly

*Equal contribution.
robust defenses fail to be robust. However, recently Tramèr et al. [127] observe that although several recently published defenses have tried to perform adaptive evaluations, many of them could still be broken by new adaptive attacks. We observe that there are repeating patterns in many of these defenses that prevent standard attacks from succeeding. This motivates us to impose restrictions on the defenses we consider in our proposed benchmark, RobustBench, which aims at \textit{standardized} adversarial robustness evaluation. Specifically, we rule out (1) classifiers which have zero gradients with respect to the input \cite{13, 48}, (2) randomized classifiers \cite{145, 91}, and (3) classifiers that contain an optimization loop in their predictions \cite{107, 76}. Often, non-certified defenses that violate these three principles only make gradient-based attacks harder but do not substantially improve adversarial robustness \cite{16}. We start from benchmarking robustness with respect to the $\ell_\infty$- and $\ell_2$-threat models, since they are the most studied settings in the literature. We use the recent AutoAttack \cite{26} as our current standard evaluation which is an ensemble of diverse parameter-free attacks (white- and black-box) that has shown for various datasets reliable performance over a large set of models that satisfy our restrictions. Moreover, we accept evaluations based on adaptive attacks whenever they can improve our standard evaluation. Additionally, we collect models robust against common image corruptions \cite{53} as these represent another type of perturbations which should not modify the decision of a classifier although they are not produced in an adversarial way.

\textbf{Contributions.} We make the following contributions with our RobustBench benchmark:

- **Leaderboard** \url{https://robustbench.github.io/}: a website with the leaderboard (see Fig. 1) based on more than 90 models where it is possible to track the progress and the current state of the art in adversarial robustness based on a standardized evaluation using AutoAttack (potentially complemented by adaptive attacks). The goal is to clearly identify the most successful ideas in training robust models to accelerate the progress in the field.

- **Model Zoo** \url{https://github.com/RobustBench/robustbench}: a collection of the most robust models that are easy to use for any downstream applications. For example, we expect that this will foster the development of better adversarial attacks by making it easier to perform evaluations on a large set of more than 60 models.

- **Analysis**: based on the collected models from the Model Zoo, we provide an analysis of how robustness affects the performance on distribution shifts, calibration, out-of-distribution detection, fairness, privacy leakage, smoothness, and transferability. In particular, we find that robust models are significantly \textit{underconfident} that leads to worse calibration, and that not all robust models have higher privacy leakage than standard models.

We believe that our standardized benchmark and accompanied collection of models will accelerate progress on multiple fronts in the area of adversarial robustness.

\section{Background and related work}

\textbf{Adversarial perturbations.} Let $x \in \mathbb{R}^d$ be an input point and $y \in \{1, \ldots, C\}$ be its correct label. For a classifier $f : \mathbb{R}^d \rightarrow \mathbb{R}^C$, we define a \textit{successful adversarial perturbation} with respect to the perturbation set $\Delta \subseteq \mathbb{R}^d$ as a vector $\delta \in \mathbb{R}^d$ such that

$$\arg \max_{c \in \{1, \ldots, C\}} f(x + \delta)_c \neq y \quad \text{and} \quad \delta \in \Delta,$$

where typically the perturbation set $\Delta$ is chosen such that all points in $x + \delta$ have $y$ as their true label. This motivates a typical robustness measure called \textit{robust accuracy}, which is the fraction of
datapoints on which the classifier $f$ predicts the correct class for all possible perturbations from the set $\Delta$. Computing the exact robust accuracy is in general intractable and, when considering $\ell_p$-balls as $\Delta$, NP-hard even for single-layer neural networks [64, 134]. In practice, an upper bound on the robust accuracy is computed via some adversarial attacks which are mostly based on optimizing some differentiable loss (e.g., cross entropy) using local search algorithms like projected gradient descent (PGD) in order to find a successful adversarial perturbation. The tightness of the upper bound depends on the effectiveness of the attack: unsuitable techniques or suboptimal parameters (e.g., the step size and the number of iterations) can make the models appear more robust than they actually are [34, 86], especially in the presence of phenomena like gradient obfuscation [5]. Certified methods [136, 44] instead provide lower bounds on robust accuracy but often underestimate robustness significantly, in particular if the certification was not part of the training process. Thus, we do not consider lower bounds in our benchmark, and focus only on upper bounds which are typically much tighter [124].

**Threat models.** We focus on the fully white-box setting, i.e. the model $f$ is assumed to be fully known to the attacker. The threat model is defined by the set $\Delta$ of the allowed perturbations: the most widely studied ones are the $\ell_p$-perturbations, i.e. $\Delta_p = \{ \delta \in \mathbb{R}^d, \|\delta\|_p \leq \varepsilon \}$, particularly for $p = \infty$ [121, 42, 79]. We rely on thresholds $\varepsilon$ established in the literature which are chosen such that the true label should stay the same for each in-distribution input within the perturbation set. We note that robustness towards small $\ell_p$-perturbations is a necessary but not sufficient notion of robustness which has been criticized in the literature [41]. It is an active area of research to develop threat models which are more aligned with the human perception such as spatial perturbations [39, 37], Wasserstein-bounded perturbations [137, 57], perturbations of the image colors [72] or $\ell_p$-perturbations in the latent space of a neural network [73, 135]. However, despite the simplicity of the $\ell_p$-perturbation model, it has numerous interesting applications that go beyond security considerations [126, 105] and span transfer learning [106, 130], interpretability [128, 65, 36], generalization [142, 156, 9], robustness to unseen perturbations [62, 142, 73, 67], stabilization of GAN training [155]. Thus, improvements in $\ell_p$-robustness have the potential to improve many of these downstream applications.

**Common corruptions.** Unlike adversarial perturbations, common corruptions [53] try to mimic modifications of the input images which can occur naturally: they are not imperceptible and evaluation on them is done in the average case fashion, i.e. there is no attacker who aims at changing the classifier’s decision. In this case, the robustness of a model is evaluated as classification accuracy on the corrupted images, averaged over types and severities of corruptions.

**Related libraries and benchmarks.** There are many libraries that focus primarily on implementations of popular adversarial attacks such as FoolBox [99], Cleverhans [95], AdverTorch [31], AdvBox [43], ART [89], SecML [83]. Some of them also provide implementations of several basic defenses, but they do not include up-to-date state-of-the-art models.

The two challenges [71, 10] hosted at NeurIPS 2017 and 2018 aimed at finding the most robust models for specific attacks, but they had a predefined deadline, so they could capture the best defenses only at the time of the competition. Ling et al. [77] proposed DEEPSEC, a benchmark that tests many combinations of attacks and defenses, but suffers from a few shortcomings as suggested by Carlini [15], in particular: (1) reporting average-case performance over multiple attacks instead of worst-case performance, (2) evaluating robustness in threat models different from the one used for training, (3) using excessively large perturbations.

Recently, Dong et al. [33] have provided an evaluation of a few defenses (in particular, 3 for $\ell_\infty$- and 2 for $\ell_2$-norm on CIFAR-10) against multiple commonly used attacks. However, they did not include some of the best performing defenses [55, 18, 46, 100] and attacks [45, 25], and in a few cases, their evaluation suggests robustness higher than what was reported in the original papers. Moreover, they do not impose any restrictions on the models they accept to the benchmark. RobustML (https://www.robust-ml.org/) aims at collecting robustness claims for defenses together with external evaluations. Their format does not assume running any baseline attack, so it relies entirely on evaluations submitted by the community, which however do not occur often enough. Thus even though RobustML has been a valuable contribution to the community, now it does not provide a comprehensive overview of the recent state of the art in adversarial robustness.

Finally, it has become common practice to test new attacks wrt $\ell_\infty$ on the publicly available models from Madry et al. [79] and Zhang et al. [152], since those represent widely accepted defenses which have stood many thorough evaluations. However, having only two models per dataset (MNIST and...
CIFAR-10) does not constitute a sufficiently large testbed, and, because of the repetitive evaluations, some attacks may already overfit to those defenses.

**What is different in RobustBench.** Learning from these previous attempts, RobustBench presents a few different features compared to the aforementioned benchmarks: (1) a baseline worst-case evaluation with an ensemble of strong, standardized attacks [26] which includes both white- and black-box attacks that can be optionally extended by adaptive evaluations, (2) clearly defined threat models that correspond to the ones used during training for submitted defenses, (3) evaluation of not only standard defenses [79] but also of more recent improvements such as [18, 46, 100], (4) the Model Zoo that provides convenient access to the 60+ most robust models from the literature which can be used for downstream tasks and facilitate the development of new standardized attacks. Moreover, RobustBench is designed as an open-ended benchmark that keeps an up-to-date leaderboard, and we welcome contributions of new defenses and evaluations of adaptive attacks for particular models.

3 **Description of RobustBench**

In this section, we start by providing a detailed layout of our proposed leaderboard for $\ell_{\infty}$, $\ell_2$, and the common corruptions threat models. Next, we present the Model Zoo, which provides unified access to most networks from our leaderboards.

3.1 **Leaderboard**

**Restrictions.** We argue that benchmarking adversarial robustness in a standardized way requires some restrictions on the type of considered models. The goal of these restrictions is to prevent submissions of defenses that cause some standard attacks to fail without actually improving robustness. Specifically, we consider only classifiers $f : \mathbb{R}^d \rightarrow \mathbb{R}^C$ that

- have in general non-zero gradients with respect to the inputs. Models with zero gradients, e.g., that rely on quantization of inputs [13, 48], make gradient-based methods ineffective thus requiring zeroth-order attacks, which do not perform as well as gradient-based attacks. Alternatively, specific adaptive evaluations, e.g. with Backward Pass Differentiable Approximation [5], can be used which, however, can hardly be standardized. Moreover, we are not aware of existing defenses solely based on having zero gradients for large parts of the input space which would achieve competitive robustness.

- have a fully deterministic forward pass. To evaluate defenses with stochastic components, it is a common practice to combine standard gradient-based attacks with Expectation over Transformations [5]. While often effective, it might be not sufficient, as shown by Tramèr et al. [127]. Moreover, the classification decision of randomized models may vary over different runs for the same input, hence even the definition of robust accuracy differs from that of deterministic networks. We also note that randomization can be useful for improving robustness and deriving robustness certificates [74, 23], but it also introduces variance in the gradient estimators (both white- and black-box) which can make attacks much less effective.

- do not have an optimization loop in the forward pass. This makes backpropagation through the classifier very difficult or extremely expensive. Usually, such defenses [107, 76] need to be evaluated adaptively with attacks considering jointly the loss of the inner loop and the standard classification task.

Some of these restrictions were also discussed by [12] for the warm-up phase of their challenge. We refer the reader to Appendix E therein for an illustrative example of a trivial defense that bypasses gradient-based and some of the black-box attacks they consider.

**Overall setup.** We set up leaderboards for the $\ell_{\infty}$, $\ell_2$ and common corruption threat models on CIFAR-10 and CIFAR-100 [69] datasets (see Table 1 for details). We use the fixed budgets of $\varepsilon_{\infty} = 8/255$ and $\varepsilon_2 = 0.5$ for the $\ell_{\infty}$ and $\ell_2$ leaderboards. Most of the models shown there are taken from papers published at top-tier machine learning and computer vision conferences as shown in Fig. 2 (left). For each entry we report the reference to the original paper, standard and robust accuracy under the specific threat model (see the next paragraph for details), network architecture, venue where the paper appeared and possibly notes regarding the model. We also highlight when extra data (usually, the dataset introduced by Carmon et al. [18]) is used since it gives a clear advantage for both
clean and robust accuracy. Moreover, the leaderboard allows to search the entries by their metadata (such as title, architecture, venue) which can be useful to compare different methods that use the same architecture or to search for papers published at some recent conference.

**Evaluation of defenses.** The evaluation of robust accuracy on common corruptions [53] involves simply computing the average accuracy on corrupted images over different corruption types and severity levels.\(^1\) To evaluate robustness of \(\ell_\infty\) and \(\ell_2\) defenses, we currently use AutoAttack [26]. It is an ensemble of four attacks: a variation of PGD attack with automatically adjusted step sizes, with (1) the cross entropy loss and (2) the difference of logits ratio loss, which is a rescaling-invariant margin-based loss function, (3) the targeted version of the FAB attack [25], which minimizes the \(\ell_p\)-norm of the perturbations, and (4) the black-box Square Attack [4]. We choose AutoAttack as it includes both black-box and white-box attacks, does not require hyperparameter tuning (in particular, the step size), and consistently improves the results reported in the original papers for almost all the models (see Fig. 2 (middle)). If in the future some new standardized and parameter-free attack is shown to consistently outperform AutoAttack on a wide set of models given a similar computational cost, we will adopt it as standard evaluation. In order to verify the reproducibility of the results, we perform the standardized evaluation independently of the authors of the submitted models. Below we show an example of how one can use our library to easily benchmark a model (either external one or taken from the Model Zoo):

```python
from robustbench.eval import benchmark
clean_acc, robust_acc = benchmark(model, dataset='cifar10', threat_model='Linf')
```

Moreover, in Appendix C we also show the variability of the robust accuracy given by AutoAttack over random seeds and report its runtime for a few models from different threat models. We also accept evaluations of the individual models on the leaderboard based on adaptive or external attacks to reflect the best available upper bound on the true robust accuracy. For example, Gowal et al. [46] and Rebuffi et al. [100] evaluate their models with a hybrid of AutoAttack and MultiTargeted attack [45], that in some cases report slightly lower robust accuracy than AutoAttack alone. We reflect all such additional evaluations in our leaderboard.

**Adding new defenses.** We believe that the leaderboard is only useful if it reflects the latest advances in the field, so it needs to be constantly updated with new defenses. We intend to include evaluations of new techniques and we welcome contributions from the community which can help to keep the benchmark up-to-date. We require new entries to (1) satisfy the three restrictions stated above, (2) to be accompanied by a publicly available paper (e.g., an arXiv preprint) describing the technique used to achieve the reported results, and (3) share the model checkpoints (not necessarily publicly). We also allow *temporarily* adding entries without providing checkpoints given that the authors evaluate their models with AutoAttack. However, we will mark such evaluations as *unverified*, and to encourage reproducibility, we reserve the right to remove an entry later on if the corresponding model checkpoint is not provided. It is possible to add a new defense to the leaderboard and (optionally) the Model Zoo by opening an issue with a predefined template in our repository [https://github.com/RobustBench/robustbench](https://github.com/RobustBench/robustbench), where more details about new additions can be found.

---

\(^1\) A breakdown over corruptions and severities is also available, e.g. for CIFAR-10 models see: [https://github.com/RobustBench/robustbench/blob/master/model_info/cifar10/corruptions/unaggregated_results.csv](https://github.com/RobustBench/robustbench/blob/master/model_info/cifar10/corruptions/unaggregated_results.csv)
Table 1: The total number of models in the Model Zoo and leaderboards per dataset and threat model.

<table>
<thead>
<tr>
<th>Threat model</th>
<th>CIFAR-10</th>
<th>CIFAR-100</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\ell_\infty$ with $\varepsilon_\infty = 8/255$</td>
<td>33</td>
<td>12</td>
</tr>
<tr>
<td>$\ell_2$ with $\varepsilon_2 = 0.5$</td>
<td>14</td>
<td>4</td>
</tr>
<tr>
<td>Common corruptions [53]</td>
<td>7</td>
<td>2</td>
</tr>
</tbody>
</table>

3.2 Model Zoo

We collect the checkpoints of many networks from the leaderboard in a single repository hosted at https://github.com/RobustBench/robustbench after obtaining the permission of the authors (see Appendix A for the information on the licenses). The goal of this repository, the Model Zoo, is to make the usage of robust models as simple as possible to facilitate various downstream applications and analyses of general trends in the field. In fact, even when the checkpoints of the proposed method are made available by the authors, it is often time-consuming and not straightforward to integrate them in the same framework because of many factors such as small variations in the architectures, custom input normalizations, etc. For simplicity of implementation, at the moment we include only models implemented in PyTorch [96]. Below we illustrate how a model can be automatically downloaded and loaded via its identifier and threat model within two lines of code:

```python
from robustbench.utils import load_model
defense = load_model(model_name='Carmon2019Unlabeled',
                     dataset='cifar10', threat_model='Linf')
```

At the moment, all models (see Table 1 and Appendix E for details) are variations of ResNet [50] and WideResNet architectures [148] of different depth and width. We include the most robust models, e.g. those from Rebuffi et al. [100], but there are also defenses which pursue additional goals alongside adversarial robustness at the fixed threshold we use: e.g., Sehwag et al. [111] consider networks which are robust and compact, Wong et al. [138] focus on computationally efficient adversarial training, Ding et al. [32] aim at input-adaptive robustness as opposed to robustness within a single $\ell_p$-radius. All these factors have to be taken into account when comparing different techniques, as they have a strong influence on the final performance.

A testbed for new attacks. Another important use case of the Model Zoo is to simplify comparisons between different adversarial attacks on a wide range of models. First, the leaderboard already serves as a strong baseline for new attacks. Second, as mentioned above, new attacks are often evaluated on the models from Madry et al. [79] and Zhang et al. [152], but this may not provide a representative picture of their effectiveness. For example, currently the difference in robust accuracy between the first and second-best attacks in the CIFAR-10 leaderboard of Madry et al. [79] is only 0.03%, and between the second and third is 0.04%. Thus, we believe that a more thorough comparison should involve multiple models to prevent overfitting of the attack to one or two standard robust defenses.

4 Analysis

With unified access to multiple models from the Model Zoo, one can easily compute various performance metrics to see general trends. In the following we analyze various aspects of robust classifiers, reporting results mostly for $\ell_\infty$-robust models on CIFAR-10 while the results for other threat models and datasets can be found in Appendix D.

Progress on adversarial defenses. In Fig. 2, we plot a breakdown over conferences, the amount of robustness overestimation reported in the original papers, and we also visualize the robustness-accuracy trade-off for the $\ell_\infty$-models from the Model Zoo. First, we observe that for multiple published defenses, the reported robust accuracy is highly overestimated. We also find that the use of extra data is able to alleviate the robustness-accuracy trade-off as suggested in previous works [98]. However, so far all models with high robustness to perturbations of $\ell_\infty$-norm up to $\varepsilon = 8/255$ still suffer from noticeable degradation in clean accuracy compared to standardly trained models. Finally, it is interesting to note that the best entries of the $\ell_p$-leaderboards are still variants of PGD adversarial training [79, 152] but with various enhancements (extra data, early stopping, weight averaging).
Performance across various distribution shifts. Here we test the performance of the models from the Model Zoo on different distribution shifts ranging from common image corruptions (CIFAR-10-C, [53]) to dataset resampling bias (CIFAR-10.1, [101]) and image source shift (CINIC-10, [29]). For each of these datasets, we measure standard accuracy, and Fig. 3 shows that improvement in the robust accuracy (which often comes with an improvement in standard accuracy) on CIFAR-10 also correlates with an improvement in standard accuracy across distributional shifts. On CIFAR-10-C, we observe that robust models (particularly with respect to the $\ell_2$-norm) tend to give a significant improvement which agrees with the findings from the previous literature [40]. Concurrently with our work, Taori et al. [123] also study the robustness to different distribution shifts of many models trained on ImageNet, including some $\ell_p$-robust models. Our conclusions qualitatively agree with theirs, and we hope that our collected set of models will help to provide a more complete picture. Moreover, we measure robust accuracy, in the same threat model used on CIFAR-10, using AutoAttack [26] (see Fig. 10 in Appendix D), and notice how $\ell_p$ adversarial robustness generalizes across different datasets, and a clear positive correlation between robust accuracy on CIFAR-10 and its variations.

![Figure 3: Standard accuracy of classifiers trained against $\ell_\infty$ (left), $\ell_2$ (middle), and common corruption (right) threat model respectively, from our Model Zoo on various distribution shifts.](image)

Calibration. A classifier is calibrated if its predicted probabilities correctly reflect the actual accuracy [47]. In the context of adversarial training, calibration was considered in Hendrycks et al. [56] who focus on improving accuracy on common corruptions and in Augustin et al. [7] who focus mostly on preventing overconfident predictions on out-of-distribution inputs. We instead focus on in-distribution calibration, and in Fig. 4 plot the expected calibration error (ECE) without and with temperature rescaling [49] to minimize the ECE (which is a simple but effective post-hoc calibration method, see Appendix D for details) together with the optimal temperature for a large set of $\ell_\infty$ models. We observe that most of the $\ell_\infty$ robust models are significantly underconfident since the optimal calibration temperature is less than one for most models. The only two models in Fig. 4 which are overconfident are the standard model and the model of Ding et al. [32] that aims to maximize the margin. We see that temperature rescaling is even more important for robust models since without any rescaling the ECE is as high as 70% for the model of Pang et al. [92] (and 21% on average) compared to 4% for the standard model. Temperature rescaling significantly reduces the ECE gap between robust and standard models but it does not fix the problem completely which suggests that it is worth incorporating calibration techniques also during training of robust models. For $\ell_2$ robust models, the models can be on the contrary more calibrated by default, although the improvement vanishes if temperature rescaling is applied (see Appendix D).

![Figure 4: Expected calibration error (ECE) before (left) and after (middle) temperature rescaling, and the optimal rescaling temperature (right) for the $\ell_\infty$-robust models.](image)

Out-of-distribution detection. Ideally, a classifier should exhibit high uncertainty in its predictions when evaluated on out-of-distribution (OOD) inputs. One of the most straightforward ways to extract this uncertainty information is to use some threshold on the predicted confidence where OOD inputs
are expected to have low confidence from the model [54]. An emerging line of research aims at
developing OOD detection methods in conjunction with adversarial robustness [52, 109, 7]. In
particular, Song et al. [120] demonstrated that adversarial training [79] leads to degradation in the
robustness against OOD data. We further test this observation on all $\ell_\infty$-models trained on CIFAR-10
from the Model Zoo on three OOD datasets: CIFAR-100 [69], SVHN [88], and Describable Textures
Dataset [22]. We use the area under the ROC curve (AUROC) to measure the success in the detection
of OOD data, and show the results in Fig. 5. With $\ell_\infty$ robust models, we find that compared to
standard training, various robust training methods indeed lead to degradation of the OOD detection
quality. While extra data in standard training can improve robustness against OOD inputs, it fails
to provide similar improvements with robust training. We further find that $\ell_2$ robust models have in
general comparable OOD detection performance to standard models (see Fig. 12 in Appendix), while
the model of Augustin et al. [7] achieves even better performance since their approach explicitly
optimizes both robust accuracy and worst-case OOD detection performance.

![Figure 5: Visualization of the OOD detection quality (higher AUROC is better) for the $\ell_\infty$-robust models trained on CIFAR-10 on three OOD datasets: CIFAR-100 (left), SVHN (middle), Describable Textures (right). We detect OOD inputs based on the maximum predicted confidence [54].](image)

**Fairness in robustness.** Recent works [8, 144] have noticed that robust training [79, 152] can lead
to models whose performance varies significantly across subgroups, e.g., defined by classes. We will
refer to this performance difference as *fairness*, and here we study the influence of robust training
methods on fairness. In Fig. 6 we show the breakdown of standard and robust accuracy for the $\ell_\infty$
robust models, where one can see how the achieved robustness largely varies over classes. While in
general the classwise standard and robust accuracy correlate well, the class “deer” in $\ell_\infty$-threat model
suffers a significant degradation, unlike what happens for $\ell_2$ (see Appendix D), which might indicate
that the features of such class are particularly sensitive to $\ell_\infty$-bounded attacks. Moreover, we measure
fairness with the relative standard deviation (RSD), defined as the standard deviation divided over the
average, of robust accuracy over classes for which lower values mean more uniform distribution and
higher robustness. We observe that better robust accuracy generally leads to lower RSD values which
implies that the disparity among classes is reduced. However, some training techniques like MART
[133] can noticeably increase the RSD and thus *increase the disparity* compared to other methods
which achieve similar robustness (around 57%).

![Figure 6: Fairness of $\ell_\infty$-robust models. Left: classwise standard (dotted lines) and robust (solid) accuracy. Right: relative standard deviation (RSD) of robust accuracy over classes vs its average.](image)

**Privacy leakage.** Deep neural networks are prone to memorizing training data [115, 17]. Recent
works have highlighted that robust training exacerbates this problem [119]. Here we benchmark
privacy leakage of training data across robust networks (Fig. 7). We calculate membership inference
accuracy using output confidence of adversarial images from the training and test sets (see Appendix D
for more details). It measures how accurately we can infer whether a sample was present in the
training dataset. Our analysis reveals mixed trends. First, our results show that not all robust models
have a significantly higher privacy leakage than a standard model. We find that the inference accuracy
across robust models has a large variation, where some models even have lower privacy leakage than a standard model. It also does not have a strong correlation with the robust accuracy. In contrast, it is largely determined by the generalization gap, as using classification confidence information does not lead to a much higher inference accuracy than the baseline determined by the generalization gap (as shown in Fig. 7 (right)). Thus one can expect lower privacy leakage in robust networks as multiple previous works have explicitly aimed to reduce the generalization gap in robust training using techniques such as early stopping [102, 152, 46]. It further suggests that reducing the generalization gap in robust networks can further reduce privacy leakage.

![Figure 7: Privacy leakage of \( \ell_\infty \)-robust models. We measure privacy leakage of training data in robust networks and compare it with robust accuracy (left) and generalization gap (right).](image)

Extra experiments. In Appendix D, we show extra experiments related to the points analyzed above and describe some of the implementation details. Also, we study how adversarial perturbations transfer between different models. We find that adversarial examples strongly transfer from robust to robust, non-robust to robust, and non-robust to non-robust networks. However, we observe poor transferability of adversarial examples from robust to non-robust networks. Moreover, since prior works [51, 146] connected higher smoothness with better robustness, we analyze the smoothness of the models both at intermediate and output layers. This confirms that, for a fixed architecture, standard training yields classifiers that are significantly less smooth than robust ones. This illustrates that one can use the collected models to study the internal properties of robust networks as well.

5 Outlook

Conclusions. We believe that a standardized benchmark with clearly defined threat models, restrictions on submitted models, and tight upper bounds on robust accuracy can be useful to show which ideas in training robust models are the most successful. Recent works have already referred to our leaderboard [68, 147, 80, 122, 143], in particular as reflecting the current state of the art [100, 75, 94], and used the networks of our Model Zoo to test new adversarial attacks [83, 104, 38, 108], to evaluate test-time defenses [131] or to evaluate perceptual distances derived from them [61]. Additionally, we have shown that unified access to a large and up-to-date set of robust models can be useful to analyze multiple aspects related to robustness. First, one can easily analyze the progress of adversarial defenses over time including the amount of robustness overestimation and the robustness-accuracy tradeoff. Second, one can conveniently study the impact of robustness on other performance metrics such as accuracy under distribution shifts, calibration, out-of-distribution detection, fairness, privacy leakage, smoothness, and transferability. Overall, we think that the community has to develop a better understanding of how different types of robustness affect other aspects of the model performance and RobustBench can help to achieve this goal.

Future plans. Our intention in the future is to keep the current leaderboards up-to-date (see the maintenance plan in Appendix B) and add new leaderboards for other datasets (in particular, for ImageNet [30]) and other threat models which become widely accepted in the community. We see as potential candidates (1) sparse perturbations, e.g. bounded by \( \ell_0 \), \( \ell_1 \)-norm or adversarial patches [11, 24, 84, 27], (2) multiple \( \ell_p \)-norm perturbations [125, 81], (3) adversarially optimized common corruptions [62, 63], (4) a broad set of perturbations unseen during training [73]. Another possible direction of development of the benchmark is including defenses based on some form of test-time adaptation [114, 131], which do not fulfill the third restriction (no optimization loop). However, since those are showing promising results and drawing attention from the community, one can introduce a separate leaderboard with specific rules and evaluation protocol for them.
References


Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper’s contributions and scope? [Yes] We clearly describe all aspects of the benchmark in Sec. 3, Sec. 4 and in the Appendix.
   (b) Did you describe the limitations of your work? [Yes] We discuss that we only perform a standardized evaluation and do not evaluate models against adaptive attacks, although we accept third-party evaluations based on adaptive attacks as mentioned in Sec. 3.1. Also, our current set of leaderboards can be seen as a limitation, so in Sec. 5 we describe how we plan to expand the benchmark to new threat models and datasets.
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
(d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]

2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]

3. If you ran experiments (e.g., for benchmarks)...  
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See Sec. 3.2 and the Model Zoo page.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [N/A]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See Sec. C.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] We give in Sec. C an example of the computational time and infrastructure used to run AutoAttack on several models. However, we could not provide such details for all models in the benchmark since those where collected over time with different resources.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] See Sec. E, and in the text we cite the authors of the datasets and algorithms we use.
   (b) Did you mention the license of the assets? [Yes] See Sec. A.
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] We provide code and data in the Model Zoo page.
   (d) Did you discuss whether and how consent was obtained from people whose data you’re using/curating? [Yes] See Sec. 3.2 and Sec. A.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]