A Broader impact

We note that the restrictions we impose on the defenses allowed in our benchmark could lead to a potential bias of the community which discourages research in certain directions. It is certainly not our goal to discourage research in directions which violate restrictions of the benchmark. However, without these restrictions a reliable evaluation of adversarial robustness is not feasible and a reliable evaluation of adversarial robustness in order to identify true advances in the field is key for further progress. Thus we think that these restrictions are unavoidable for a benchmark but we are working on relaxing the restrictions as much as possible.

Additionally, in motivating higher robustness against adversarial examples, our work may leave an unwanted side effect on tasks where adversarial attacks can actually be used for beneficial purposes [76, 125, 109]. However, this is true for any paper that aims at improving adversarial robustness (either directly or indirectly via, e.g., a standardized benchmark).

On the positive side, in our work, we do not only perform a standardized benchmarking of adversarial robustness but also analyze multiple other properties of robust models such as calibration, privacy leakage, fairness, etc. In our opinion, such analyses are important since they allow us to assess the broader impact of improving robustness on other crucial performance metrics of neural networks.

Finally, we note that a good performance on our benchmark does not guarantee the safety of the benchmarked model in a real-world deployment which is likely to require more domain-specific threat models. ℓ_p-bounded adversarial attacks can be a realistic threat model in applications where it is possible to input an image directly in a digital format [141, 116]. However, attacks in-the-wild [78, 40] are usually much more involved and differ considerably from the presented simple ℓ_p-perturbations. Moreover, the common corruptions we used for evaluation from Hendrycks and Dietterich [58] are artificially generated, and thus may differ from the corruptions encountered in the real world. Taking this into account, we suggest to always think critically about the robustness requirements that are necessary for a particular application at hand.

B Licenses

The code used for benchmarking is released under MIT license. The code of AutoAttack [28] that our benchmark relies on has been released under the MIT license as well. The classifiers in the Model Zoo are added according to the permission given by the authors with the license they choose: most of the models have MIT license, other have more restrictive ones such as Attribution-NonCommercial-ShareAlike 4.0 International, Apache License 2.0, BSD 3-Clause License. The details can be found at `https://github.com/RobustBench/robustbench/blob/master/LICENSE`. The CIFAR-10 and CIFAR-100 datasets [75] are obtained via the PyTorch loaders [105], while CIFAR-10-C and CIFAR-100-C [58], with the common corruptions, are downloaded from the official release (see `https://zenodo.org/record/2535967#.YLYf9agzaUk` and `https://zenodo.org/record/3555552#.YLYeJagzaUk`). The validation set of ImageNet is not hosted or downloaded by our provided evaluation code, but it needs to be downloaded in advance directly by the user.

C Maintenance plan

Here we discuss the main aspects of maintaining `RobustBench` and the costs associated with it:

- **Hosting the website** (`https://robustbench.github.io/`): we host our leaderboard using GitHub pages[4] which is a free service.
- **Hosting the library** (`https://github.com/RobustBench/robustbench`): the code of our library is hosted on GitHub[5] which offers the basic features that we need to maintain the library for free.

---

[4] `https://pages.github.com/`

[5] `https://github.com/`

- **Hosting the models**: to ensure the availability of the models from the Model Zoo, we host them in our own cloud storage on Google Drive[6]. At the moment, they take around 24 GB of space which fits into the 100 GB storage plan that costs 2 USD per month.
- **Running evaluations**: we run all evaluations on the GPU servers that are available to our research groups which incurs no extra costs.

Moreover, as we mention in the outlook (Sec. 5), we also plan to expand the benchmark to new datasets and threat models which can slightly increase the required maintenance costs since we may need to upgrade the storage plan. We also expect the benchmark to be community-driven and to encourage this we have provided instructions[7] on how to submit new entries to the leaderboard and to the Model Zoo.

D Details of the ImageNet leaderboards

Extending the benchmark to ImageNet presents some challenges compared to CIFAR-10 and CIFAR-100. First, the ImageNet validation set (usually used as the test set) contains *50'000* images which makes it infeasible to run expensive evaluations on it. Thus, we define a fixed subset (5'000 randomly sampled images in our case) for faster evaluation, whose image IDs we make available in the Model Zoo. Second, it is not obvious how to handle the fact that different models may use different preprocessing techniques (e.g., different resolution, cropping, etc) which makes the search space for an attack *not fully comparable* across defenses. For this, we decide to allow models with different preprocessing steps and input resolution, considering them yet another design choice similarly to the choice of the network architecture which also has a large influence on the final results. Since in the ℓ_∞-threat model the constraints are componentwise independent, we use the same threshold $\varepsilon_\infty = 4/255$ for every classifier, regardless of the input dimensionality which is used after preprocessing.

E Reproducibility and runtime

Here we discuss the main aspects of the reproducibility of the benchmark.

First of all, the code to run the benchmark on a given model is available in our repository, and an example of how to run it is given in the README file. The installation instructions are also provided in the README file and the requirements will be installed automatically.

To satisfy other points from the reproducibility checklist[8] which are applicable to our benchmark, we also discuss next the variability of the robust accuracy over random seeds and the average runtime of the benchmark. Evaluation of the accuracy on common corruptions [58] is deterministic if we do not take into account non-deterministic operations on computational accelerators such as GPUs[9] which, however, do not affect the resulting accuracy. On the other hand, robustness evaluation using AutoAttack has an element of randomness since it relies on random initialization of the starting points and also on the randomness in the update of the Square Attack [4]. To show the effect of randomness on the robust accuracy given by AutoAttack, we repeat evaluation over four random seeds on four models available in the Model Zoo from different threat models covering all datasets considered. In Table 2, we report the average robust accuracy with its standard deviation and observe that different seeds lead to very similar results. Moreover, we indicate the runtime of each evaluation, which is largely influenced by the size of the model, the computing infrastructure (every run uses a single Tesla V100 GPU), and the dataset. Moreover, less robust models require less time for evaluation which is due to the fact that AutoAttack does not further attack a point if an adversarial example is already found by some preceding attack in the ensemble.

Additionally, as mentioned above, for ImageNet we have randomly sampled and fixed 5000 images from the validation set. We provide the IDs of those images and code to load them in our repository. Note that we use the same set of images for ℓ_p-robustness and for common corruptions, in which case for every point 15 types of corruptions at 5 severity levels are applied, consistently with the other datasets.

---

[6] `https://www.google.com/drive/`

[7] `https://github.com/RobustBench/robustbench#adding-a-new-model`

[8] `https://www.cs.mcgill.ca/~jpineau/ReproducibilityChecklist.pdf`

[9] `https://pytorch.org/docs/stable/notes/randomness.html`

Finally, when we extended the benchmark to ImageNet, we noticed that different versions of `PyTorch` and `torchvision` may lead to small differences in the standard accuracy (up to 0.16% on 5000 points for the same model). We suspect this is due to minor variations in the implementation of the preprocessing functions (such as resizing). Thus, we fix in the requirements `torch==1.7.1` and `torchvision==0.8.2` to ensure reproducibility. Note that the overall *ranking* and level of robustness of the defenses should not be influenced by using different versions of these libraries. We have not noticed similar issues for the other datasets.

Table 2: Statistics about the standardized evaluation with AutoAttack when repeated for four random seeds. We can see that the robust accuracy has very small fluctuations. We also report the runtime for the different models which is much smaller for less robust models.

Dataset	**Leaderboard**	**Paper**	**Architecture**	**Clean acc.**	**Robust acc.**	**Time**
CIFAR-10	ℓ_∞	Gowal et al. [50]	WRN-28-10	89.48%	62.82% ± 0.016	11.8 h
CIFAR-10	ℓ_2	Rebuffi et al. [111]	WRN-28-10	91.79%	78.80% ± 0.000	15.1 h
CIFAR-100	ℓ_∞	Wu et al. [155]	WRN-34-10	60.38%	28.84% ± 0.018	6.6 h
ImageNet	ℓ_∞	Salman et al. [117]	ResNet-18	52.92%	25.31% ± 0.010	1.6 h

F Additional analysis

In this section, we show more results on different datasets and/or threat models and discuss some implementation details related to the analysis from Sec. 4. We also additionally analyze the *smoothness* and *transferability* properties of the models from the Model Zoo.

Progress on adversarial defenses. As done in the main part for the ℓ_∞-robust models on CIFAR-10, we show here the same statistics but for ℓ_2-robust models on CIFAR-10 in Fig. 8 and for ℓ_∞-robust models on CIFAR-100 in Fig. 9. We observe a few differences compared to the ℓ_∞-robust models on CIFAR-10 reported in Fig. 2. First of all, the amount of robustness overestimation is not large and in particular there are no models that have *zero* robust accuracy. Second, we can see that the best ℓ_2-robust models on CIFAR-10 has even higher standard accuracy than a standard model (95.74% vs 94.78%) while having a significantly higher robust accuracy (82.32% vs 0.00%) and leaving a relatively small gap between the standard and robust accuracy. Finally, we note that the progress on the ℓ_∞-threat model on CIFAR-100 is more recent and there are only a few published papers that report adversarial robustness on this dataset.

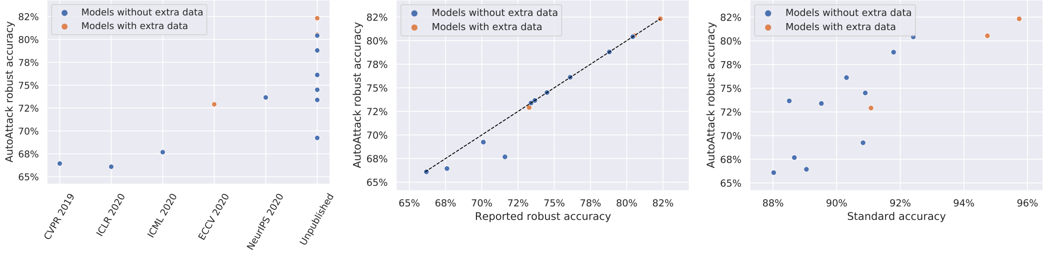


Figure 8: Visualization of the robustness and accuracy of 13 CIFAR-10 models from the `RobustBench` ℓ_2-leaderboard. Robustness is evaluated using ℓ_2-perturbations with $\varepsilon_2 = 0.5$.

Robustness across distribution shifts. We measure robust accuracy on various distribution shifts using four dataset, namely CIFAR-10, CINIC-10, CIFAR-10.1, and CIFAR-10-C. In particular, we compute the robust accuracy in the same threat model as for the original CIFAR-10 dataset, and report the results in Fig. 10. Interestingly, one can observe that ℓ_p adversarial robustness is maintained under the distribution shifts, and it highly correlates with the robustness on the dataset the models were trained on (i.e. CIFAR-10).

Calibration. We compute the expected calibration error (ECE) using the code of [52]. We use their default settings to compute the calibration error which includes, in particular, binning of the probability range onto 15 equally-sized bins. However, we use our own implementation of the temperature rescaling algorithm which is close to that of [7]. Since optimization of the ECE over the

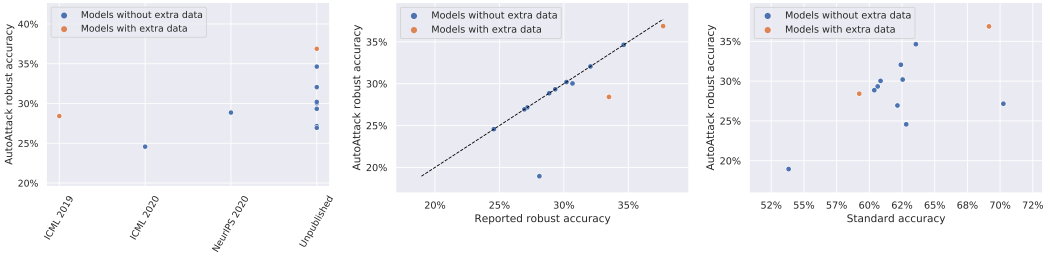


Figure 9: Visualization of the robustness and accuracy of 12 CIFAR-100 models from the `RobustBench` ℓ_∞-leaderboard. Robustness is evaluated using ℓ_∞-perturbations with $\varepsilon_\infty = 8/255$.

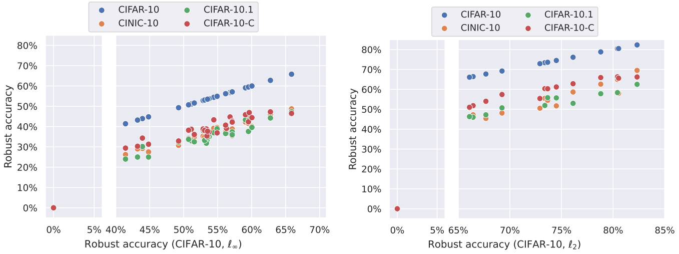


Figure 10: Robust accuracy of the robust classifiers, trained against ℓ_∞ and ℓ_2 threat model, respectively, from our Model Zoo on various distribution shifts. The data points with 0% robust accuracy correspond to a standardly trained model.

softmax temperature is a simple *one-dimensional* optimization problem, we can solve it efficiently using a grid search. Moreover, the advantage of performing a grid search is that we can optimize directly the metric of interest, i.e. ECE, instead of the cross-entropy loss as in [52] who had to rely on a differentiable loss since they used LBFGS [87] to optimize the temperature. We perform a grid search over the interval $t \in [0.001, 1.0]$ with a grid step 0.001 and we test both t and $1/t$ temperatures. Moreover, we check that for all models the optimal temperature t is situated not at the boundary of the grid.

We show additional calibration results for ℓ_2-robust models in Fig. 11. The overall trend of the ECE is the same as for ℓ_∞-robust models: most of the ℓ_2 models are underconfident (since the optimal temperature is less than one) and lead to worse calibration before and after temperature rescaling. The main difference compared to the ℓ_∞ threat model is that among the ℓ_2 models there are two models that are *better-calibrated*: one before (Engstrom et al. [37] with 1.41% ECE vs 3.71% ECE of the standard model) and one after (Gowal et al. [50] with 1.00% ECE vs 1.11% ECE of the standard model) temperature rescaling. Moreover, we can see that similarly to the ℓ_∞ case, the only overconfident models are either the standard one or models that maximize the margin instead of using norm-bounded perturbations, i.e. Ding et al. [34] and Rony et al. [114].

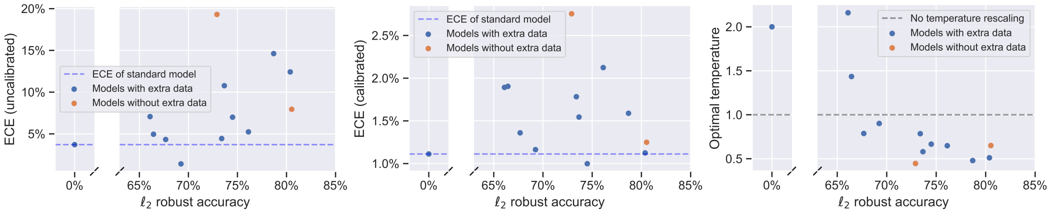


Figure 11: Expected calibration error (ECE) before (**left**) and after (**middle**) temperature rescaling, and the optimal rescaling temperature (**right**) for the ℓ_2-robust models.

Out-of-distribution detection. Fig. 12 complements Fig. 5 and shows the ability of ℓ_2-robust models trained on CIFAR-10 to distinguish inputs from other datasets (CIFAR-100, SVHN, Describable Textures). We find that ℓ_2 robust models have in general comparable OOD detection performance to standardly trained models, while the model by Augustin et al. [7] achieves even better performance since their approach explicitly optimizes both robust accuracy and worst-case OOD detection performance.

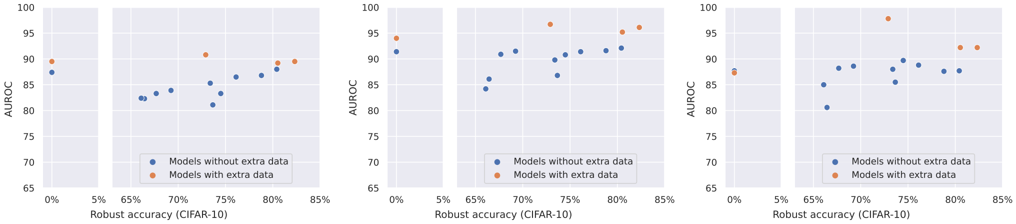


Figure 12: Visualization of the quality of OOD detection (higher AUROC is better) for the ℓ_2-robust models on three different OOD datasets: CIFAR-100 (**left**), SVHN (**middle**), Describable Textures (**right**).

Fairness in robustness. We report the results about fairness for robust models in the ℓ_2-threat model in Fig. 13, similarly to what done for ℓ_∞ above. We see that the difference in robustness among classes is similar to what observed for the ℓ_∞ models. Also, the RSD of robustness over classes decreases, which indicates that the disparity among subgroups is reduced, as the average robust accuracy improves. To compute the robustness for the experiments about fairness we used APGD on the targeted DLR loss [28] with 3 target classes and 20 iterations each on the whole test set. Note that even with this smaller budget we achieve results very close to that of the full evaluation, with an average difference smaller than 0.5%.

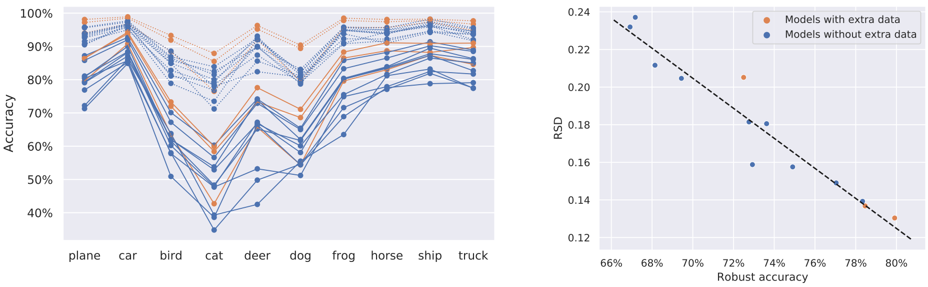
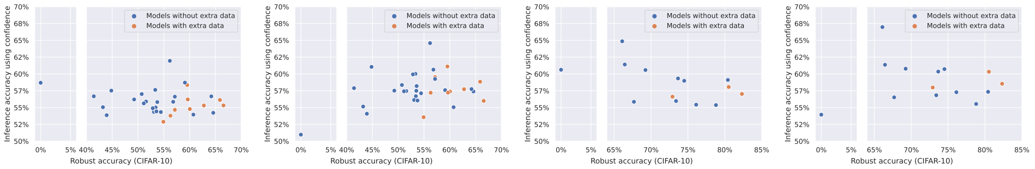


Figure 13: **Left:** classwise standard (dotted lines) and robust (solid) accuracy of ℓ_2-robust models. **Right:** relative standard deviation (RSD) of robust accuracy over classes vs its average.

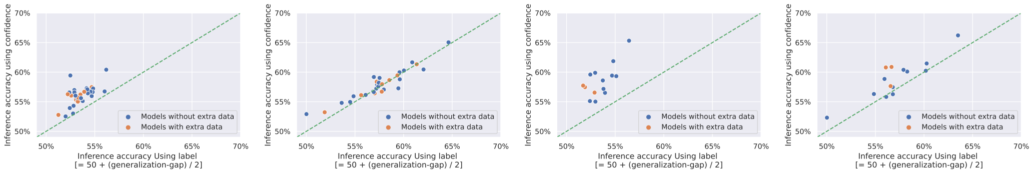
Privacy leakage. We use membership inference accuracy, referred to as inference accuracy, as a measure of the leakage of training data details from pre-trained neural networks. It measures how successfully we can identify whether a particular sample was present in the training set. We closely follow the methodology described in Song and Mittal [130] to calculate inference accuracy. In particular, we measure the confidence in the correct class for each input image with a pre-trained classifier. We measure the confidence for both training and test set images and calculate the maximum classification accuracy between train and test images based on the confidence values. We refer to this accuracy as *inference accuracy using confidence*. We also follow the recommendation from Song et al. [131] where they show that adversarial examples are more successful in estimating inference accuracy on robust networks. In our experiments, we also find that using adversarial examples leads to higher inference accuracy than benign images (Figure 14). We also find that robust networks in the ℓ_2 threat model have relatively higher inference accuracy than robust networks in the ℓ_∞ threat model.

A key reason behind privacy leakage through membership inference is that deep neural networks often end up overfitting on the training data. One standard metric to measure overfitting is the generalization gap between train and test set. Naturally, this difference in the accuracy on the train and test set is the baseline of inference accuracy. We refer to it as *inference accuracy using label* and report it in Figure 15. We consider both benign and adversarial images. When using benign images, we find confidence information does lead to higher inference accuracy than using only labels. However, with adversarial examples, which achieve higher inference accuracy than benign images, we find that inference accuracy based on confidence information closely follows the inference accuracy calculate from labels.



(a) Benign (ℓ_∞) (b) Adversarial (ℓ_∞) (c) Benign (ℓ_2) (d) Adversarial (ℓ_2)

Figure 14: **Comparing privacy leakage of different networks.** We compare membership inference accuracy from benign and adversarial images across both ℓ_∞ and ℓ_2 threat model.



(a) Benign (ℓ_∞) (b) Adversarial (ℓ_∞) (c) Benign (ℓ_2) (d) Adversarial (ℓ_2)

Figure 15: **Comparing privacy leakage with different output statistics.** We measure privacy leakage using membership inference accuracy, i.e., classification success between train and test set. We measure it using two baselines 1) based on correct prediction i.e., using predicted class label and 2) based on classification confidence in correct class. We also measure it using both benign and adversarial images.

Smoothness. Previous work [162] has shown that smoothness of a model, together with enough separation between the classes of the dataset for which it is trained, is necessary to achieve both natural and robust accuracy. They use local Lipschitzness as a measure for model smoothness, and observe empirically that robust models are more smoother than models trained in a standard way. Our Model Zoo enables us to check this fact empirically on a wider range of robust models, trained with a more diverse set of techniques, in particular with and without extra training data. Moreover, as we have access to the model internals, we can also compute local Lipschitzness of the model up to arbitrary layers, to see how smoothness changes between layers.

We compute local Lipschitzness using projected gradient descent (PGD) on the following optimization problem:

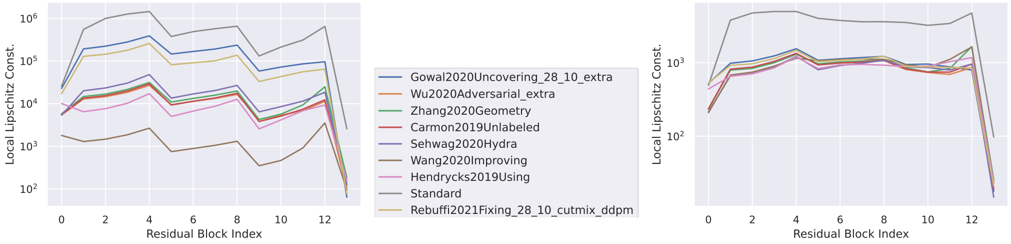
$$L = \frac{1}{N} \sum_{i=1}^{N} \max_{\substack{x_1 : \|x_1 - x_i\|_\infty \le \varepsilon, \\ x_2 : \|x_2 - x_i\|_\infty \le \varepsilon}} \frac{\|f(x_1) - f(x_2)\|_1}{\|x_1 - x_2\|_\infty}, \tag{2}$$

where x_i represents each sample around which we compute local Lipschitzness, N is the number of samples across which we average ($N = 256$ in all our experiments), and f represents the function whose Lipschitz constant we compute. As mentioned above, this function can be either the full model, or the model up to an arbitrary intermediate layer.

Since the models can have similar smoothness behavior, but at a different scale, we also consider normalizing the models outputs. One such normalization we use is given by the projection of the model outputs on the unit ℓ_2 ball. This normalization aims at capturing the angular change of the output, instead of taking in consideration also its magnitude. We compute the "angular" version of the Lipschitz constant as

$$L = \frac{1}{N} \sum_{i=1}^{N} \max_{\substack{x_1 : \|x_1 - x_i\|_\infty \le \varepsilon, \\ x_2 : \|x_2 - x_i\|_\infty \le \varepsilon}} \frac{\left\| \frac{f(x_1)}{\|f(x_1)\|_2} - \frac{f(x_2)}{\|f(x_2)\|_2} \right\|_1}{\|x_1 - x_2\|_\infty}. \tag{3}$$

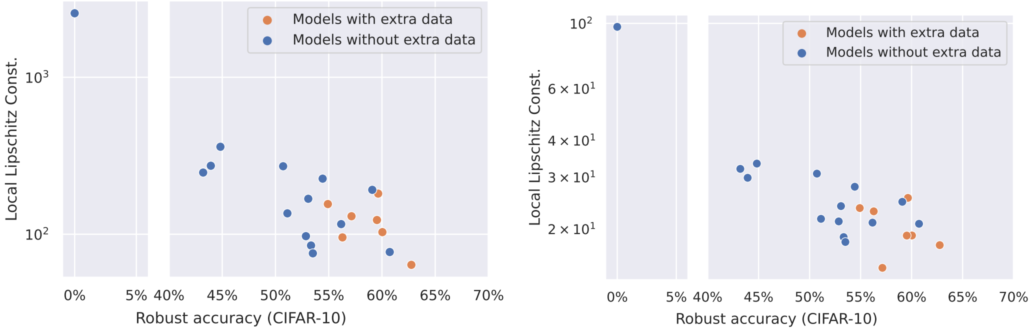
For both variations of Lipschitzness, we compute it with $\varepsilon = 8/255$, running the PDG-like procedure for 50 steps, with a step size of $\varepsilon\,/5$.



(a) No normalization

(b) ℓ_2 unit ball normalization

Figure 16: **Lipschitzness.** Computation of the local Lipschitz constant of the WRN-28-10 ℓ_∞-robust models in our Model Zoo with $\varepsilon = 8/255$. The color coding of the models is the same across both figures. For the correspondence between model IDs (shown in the legend) and papers that introduced them, see Appendix G.



(a) $\varepsilon = 8/255$, no normalization

(b) $\varepsilon = 8/255$, ℓ_2 unit ball normalization

Figure 17: **Lipschitzness vs Robustness.** Local Lipschitz constant of the output layer vs. robust accuracy of a subset of the ℓ_∞-robust models in our Model Zoo.

In Fig. 16 we compute the layerwise Lipschitzness for all ℓ_∞ models trained on CIFAR-10 from the Model Zoo that have the WRN-28-10 architecture. We observe that the standard model is the least smooth at all the layers, and that all the robustly trained models are smoother. Moreover, we can notice that in Fig. 16a there are two models in the middle ground: these are the models by Gowal et al. [50] and Rebuffi et al. [111], which are the most robust ones, up to the last layer, the smoothest. Nonetheless, in the middle layers, they are the second and third least smooth, according to the unnormalized local Lipschitzness. This can be due to the different activation function used in these models (Swish vs ReLU). For this reason, we also compute "angular" Lipschitzness according to Eq. 3. Indeed, in Fig. 16b, all the robust models are in the same order of magnitude at all layers.

Finally, we also show the Lipschitz constants of the output layer for a larger set of ℓ_∞ models from the Model Zoo that are not restricted to the same architecture. We plot the Lipschitz constant vs. the robust accuracy for these models in Fig 17. We see that there is a clear relationship between robust accuracy and Lipschitzness, hence confirming the findings of Yang et al. [162].

Transferability. We generate adversarial examples for a network, referred to as source network, and measure robust accuracy of every other network, referred to as target network, from the model zoo on them. We also include additional non-robust models[10], to name a few, VGG19, ResNet18, and DenseNet121, in our analysis. We consider both ten step PGD attack and FGSM attack to generate adversarial examples as two transferability baselines commonly used in the literature. For both attacks, we use the cross-entropy loss, and for the PGD attack we use ten iterations and step size $\varepsilon/4$.

We present our results in Figure 18, 19 where the correspondence between model IDs and papers that introduced them can be found in Appendix G. We find that transferability to each robust target network follows a similar trend where adversarial examples transfer equally well from another robust

---

[10] We train then for 200 epochs and achieve 93-95% clean accuracy for all networks on the CIFAR-10 dataset.

networks. Though slight worse than robust network, adversarial example from non-robust network also transfer equally well to robust networks. We observe a strong transferability among non-robust networks with adversarial examples generated from PGD attacks. Adversarial examples generated using the FGSM attack also transfer successfully. However, they achieve lower robust accuracy on the target network. Intriguingly, we observe the weakest transferability from a robust to a non-robust network. This observation holds for all robust source networks across both FGSM and PGD-attack in both ℓ_∞ and ℓ_2 threat model.

G Leaderboards

We here report the details of all the models included in the various leaderboards, for the ℓ_∞-, ℓ_2-threat models and common corruptions. In particular, we show for each model the clean accuracy, robust accuracy (either on adversarial attacks or corrupted images), whether additional data is used for training, the architecture used, the venue at which it appeared and, if available, the identifier in the Model Zoo (which is also used in some of the experiments in Sec. F).

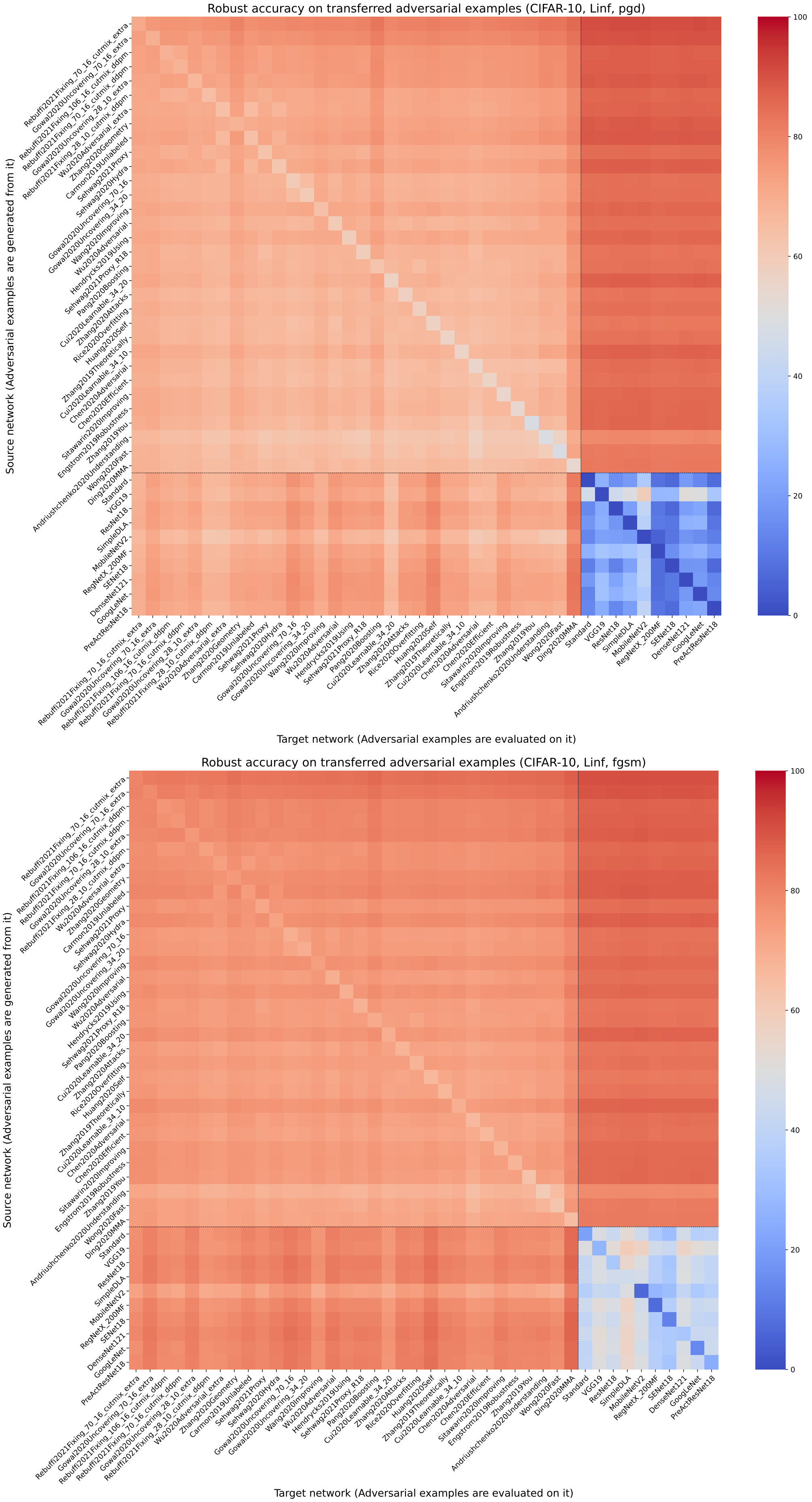
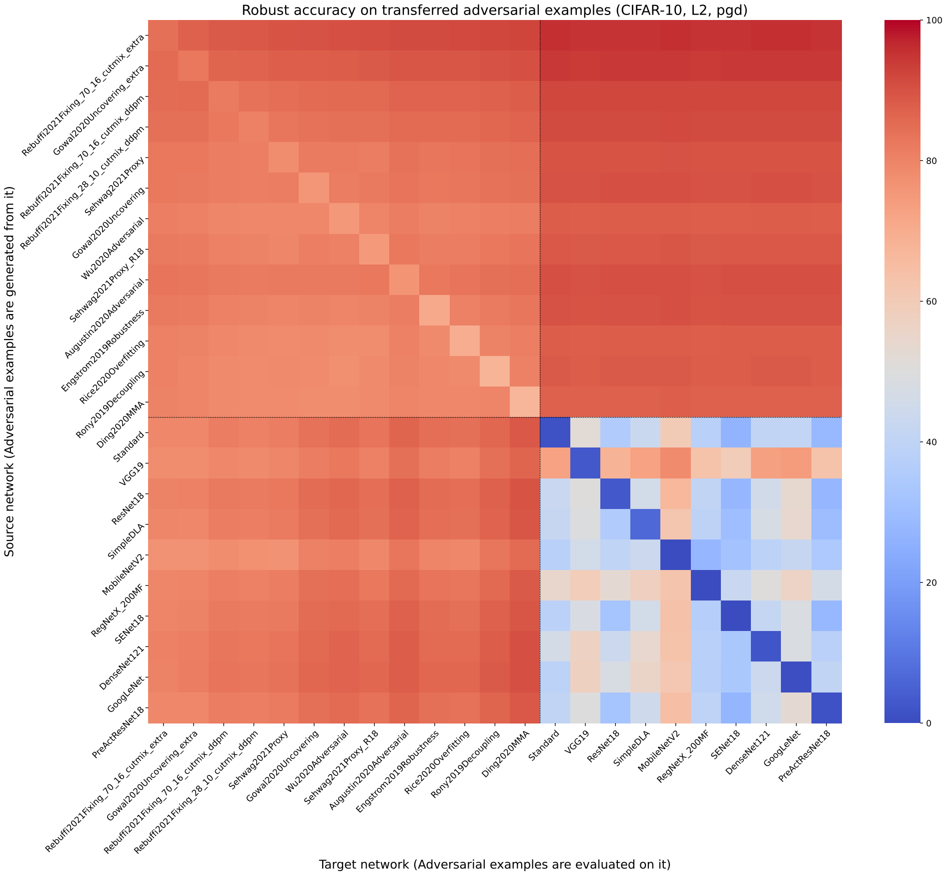


Figure 18: Measuring transferability of adversarial examples (ℓ_∞, $\epsilon = 8/255$). We use a ten step PGD attack in top figure and FGSM attack in bottom figure. Lower robust accuracy implies better transferability.



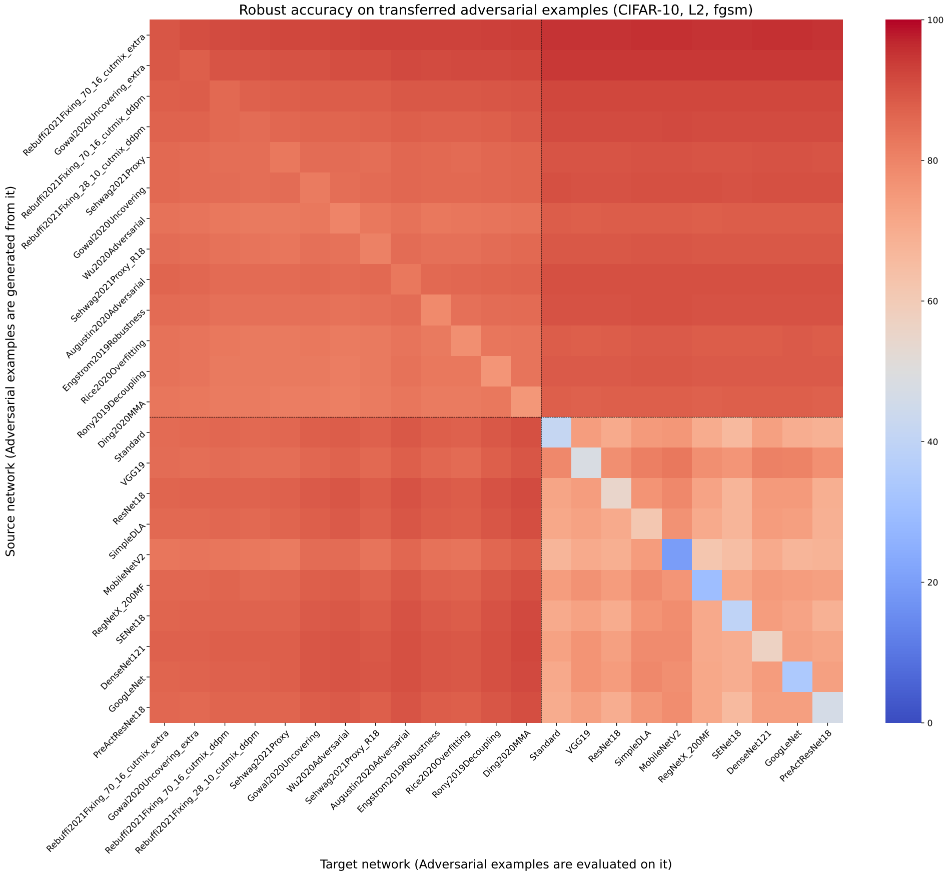


Figure 19: Measuring transferability of adversarial examples (ℓ_2, $\epsilon = 0.5$). We use a ten step PGD attack in top figure and FGSM attack in bottom figure. Lower robust accuracy implies better transferability.

Table 3: Leaderboard for the ℓ_∞-threat model, CIFAR-10.

	Model	Clean	Robust	Extra data	Architecture	Venue	Model Zoo ID
1	Rebuffi et al. [111]	92.23	66.56	Y	WRN-70-16	arXiv, Mar 2021	Rebuffi2021Fixing_70_16_cutmix_extra
2	Gowal et al. [50]	91.10	65.87	Y	WRN-70-16	arXiv, Oct 2020	Gowal2020Uncovering_70_16_extra
3	Rebuffi et al. [111]	88.50	64.58	N	WRN-106-16	arXiv, Mar 2021	Rebuffi2021Fixing_106_16_cutmix_ddpm
4	Rebuffi et al. [111]	88.54	64.20	N	WRN-70-16	arXiv, Mar 2021	Rebuffi2021Fixing_70_16_cutmix_ddpm
5	Rade and Moosavi-Dezfooli [107]	91.47	62.83	Y	WRN-34-10	OpenReview, Jun 2021	Rade2021Helper_extra
6	Gowal et al. [50]	89.48	62.76	Y	WRN-28-10	arXiv, Oct 2020	Gowal2020Uncovering_28_10_extra
7	Rade and Moosavi-Dezfooli [107]	88.16	60.97	N	WRN-28-10	OpenReview, Jun 2021	Rade2021Helper_ddpm
8	Rebuffi et al. [111]	87.33	60.73	N	WRN-28-10	arXiv, Mar 2021	Rebuffi2021Fixing_28_10_cutmix_ddpm
9	Wu et al. [154]	87.67	60.65	Y	WRN-34-15	arXiv, Oct 2020	N/A
10	Sridhar et al. [133]	86.53	60.41	Y	WRN-34-15	arXiv, Jun 2021	Sridhar2021Robust_34_15
11	Wu et al. [155]	88.25	60.04	Y	WRN-28-10	NeurIPS 2020	Wu2020Adversarial_extra
12	Sridhar et al. [133]	89.46	59.66	Y	WRN-28-10	arXiv, Jun 2021	Sridhar2021Robust
13	Zhang et al. [171]	89.36	59.64	Y	WRN-28-10	ICLR 2021	Zhang2020Geometry
14	Carmon et al. [19]	89.69	59.53	Y	WRN-28-10	NeurIPS 2019	Carmon2019Unlabeled
15	Sehwag et al. [123]	85.85	59.09	N	WRN-34-10	arXiv, Apr 2021	Sehwag2021Proxy
16	Rade and Moosavi-Dezfooli [107]	89.02	57.67	Y	PreActRN-18	OpenReview, Jun 2021	Rade2021Helper_R18_extra
17	Gowal et al. [50]	85.29	57.14	N	WRN-70-16	arXiv, Oct 2020	Gowal2020Uncovering_70_16
18	Sehwag et al. [121]	88.98	57.14	Y	WRN-28-10	NeurIPS 2020	Sehwag2020Hydra
19	Rade and Moosavi-Dezfooli [107]	86.86	57.09	N	PreActRN-18	OpenReview, Jun 2021	Rade2021Helper_R18_ddpm
20	Gowal et al. [50]	85.64	56.82	N	WRN-34-20	arXiv, Oct 2020	Gowal2020Uncovering_34_20
21	Rebuffi et al. [111]	83.53	56.66	N	PreActRN-18	arXiv, Mar 2021	Rebuffi2021Fixing_R18_ddpm
22	Wang et al. [148]	87.50	56.29	Y	WRN-28-10	ICLR 2020	Wang2020Improving
23	Wu et al. [155]	85.36	56.17	N	WRN-34-10	NeurIPS 2020	Wu2020Adversarial
24	Uesato et al. [144]	86.46	56.03	Y	WRN-28-10	NeurIPS 2019	N/A
25	Hendrycks et al. [60]	87.11	54.92	Y	WRN-28-10	ICML 2019	Hendrycks2019Using
26	Sehwag et al. [123]	84.38	54.43	N	RN-18	arXiv, Apr 2021	Sehwag2021Proxy_R18
27	Pang et al. [102]	86.43	54.39	N	WRN-34-20	ICLR 2021	N/A
28	Pang et al. [101]	85.14	53.74	N	WRN-34-20	NeurIPS 2020	Pang2020Boosting
29	Cui et al. [30]	88.70	53.57	N	WRN-34-20	ICCV 2021	Cui2020Learnable_34_20
30	Zhang et al. [170]	84.52	53.51	N	WRN-34-10	ICML 2020	Zhang2020Attacks
31	Rice et al. [113]	85.34	53.42	N	WRN-34-20	ICML 2020	Rice2020Overfitting
32	Huang et al. [64]	83.48	53.34	N	WRN-34-10	NeurIPS 2020	Huang2020Self
33	Zhang et al. [168]	84.92	53.08	N	WRN-34-10	ICML 2019	Zhang2019Theoretically
34	Cui et al. [30]	88.22	52.86	N	WRN-34-10	ICCV 2021	Cui2020Learnable_34_10
35	Qin et al. [106]	86.28	52.84	N	WRN-40-8	NeurIPS 2019	N/A
36	Chen et al. [23]	86.04	51.56	N	RN-50	CVPR 2020	Chen2020Adversarial
37	Chen et al. [22]	85.32	51.12	N	WRN-34-10	arXiv, Oct 2020	Chen2020Efficient
38	Sitawarin et al. [129]	86.84	50.72	N	WRN-34-10	arXiv, Mar 2020	Sitawarin2020Improving
39	Engstrom et al. [37]	87.03	49.25	N	RN-50	GitHub, Oct 2019	Engstrom2019Robustness
40	Singh et al. [128]	87.80	49.12	N	WRN-34-10	IJCAI 2019	N/A
41	Mao et al. [91]	86.21	47.41	N	WRN-34-10	NeurIPS 2019	N/A
42	Zhang et al. [165]	87.20	44.83	N	WRN-34-10	NeurIPS 2019	Zhang2019You
43	Madry et al. [88]	87.14	44.04	N	WRN-34-10	ICLR 2018	N/A
44	Andriushchenko and Flammarion [3]	79.84	43.93	N	PreActRN-18	NeurIPS 2020	Andriushchenko2020Understanding
45	Pang et al. [99]	80.89	43.48	N	RN-32	ICLR 2020	N/A
46	Wong et al. [153]	83.34	43.21	N	PreActRN-18	ICLR 2020	Wong2020Fast
47	Shafahi et al. [124]	86.11	41.47	N	WRN-34-10	NeurIPS 2019	N/A
48	Ding et al. [34]	84.36	41.44	N	WRN-28-4	ICLR 2020	Ding2020MMA
49	Kundu et al. [77]	87.32	40.41	N	RN-18	ASP-DAC 2021	N/A
50	Atzmon et al. [6]	81.30	40.22	N	RN-18	NeurIPS 2019	N/A
51	Moosavi-Dezfooli et al. [94]	83.11	38.50	N	RN-18	CVPR 2019	N/A
52	Zhang and Wang [166]	89.98	36.64	N	WRN-28-10	NeurIPS 2019	N/A
53	Zhang and Xu [167]	90.25	36.45	N	WRN-28-10	OpenReview, Sep 2019	N/A
54	Jang et al. [65]	78.91	34.95	N	RN-20	ICCV 2019	N/A
55	Kim and Wang [72]	91.51	34.22	N	WRN-34-10	OpenReview, Sep 2019	N/A
56	Zhang et al. [169]	44.73	32.64	N	5-layer-CNN	ICLR 2020	N/A
57	Wang and Zhang [147]	92.80	29.35	N	WRN-28-10	ICCV 2019	N/A
58	Xiao et al. [156]	79.28	7.15	N	DenseNet-121	ICLR 2020	N/A
59	Jin and Rinard [66]	90.84	1.35	N	RN-18	arXiv, Mar 2020	N/A
60	Mustafa et al. [96]	89.16	0.28	N	RN-110	ICCV 2019	N/A
61	Chan et al. [20]	93.79	0.26	N	WRN-34-10	ICLR 2020	N/A
62	Standard	94.78	0.0	N	WRN-28-10	N/A	N/A
63	Alfarra et al. [1]	91.03	0.00	N	WRN-28-10	arXiv, Jun 2020	N/A

Table 4: Leaderboard for the ℓ_2-threat model, CIFAR-10.

	Model	Clean	Robust	Extra data	Architecture	Venue	Model Zoo ID
1	Rebuffi et al. [111]	95.74	82.32	Y	WRN-70-16	arXiv, Mar 2021	Rebuffi2021Fixing_70_16_cutmix_extra
2	Gowal et al. [50]	94.74	80.53	Y	WRN-70-16	arXiv, Oct 2020	Gowal2020Uncovering_extra
3	Rebuffi et al. [111]	92.41	80.42	N	WRN-70-16	arXiv, Mar 2021	Rebuffi2021Fixing_70_16_cutmix_ddpm
4	Rebuffi et al. [111]	91.79	78.80	N	WRN-28-10	arXiv, Mar 2021	Rebuffi2021Fixing_28_10_cutmix_ddpm
5	Augustin et al. [7]	93.96	78.79	Y	WRN-34-10	ECCV 2020	Augustin2020Adversarial_34_10_extra
6	Augustin et al. [7]	92.23	76.25	Y	WRN-34-10	ECCV 2020	Augustin2020Adversarial_34_10
7	Rade and Moosavi-Dezfooli [107]	90.57	76.15	N	PreActRN-18	OpenReview, Jun 2021	Rade2021Helper_R18_ddpm
8	Sehwag et al. [123]	90.31	76.12	N	WRN-34-10	arXiv, Apr 2021	Sehwag2021Proxy
9	Rebuffi et al. [111]	90.33	75.86	N	PreActRN-18	arXiv, Mar 2021	Rebuffi2021Fixing_R18_cutmix_ddpm
10	Gowal et al. [50]	90.90	74.50	N	WRN-70-16	arXiv, Oct 2020	Gowal2020Uncovering
11	Wu et al. [155]	88.51	73.66	N	WRN-34-10	NeurIPS 2020	Wu2020Adversarial
12	Sehwag et al. [123]	89.52	73.39	N	RN-18	arXiv, Apr 2021	Sehwag2021Proxy_R18
13	Augustin et al. [7]	91.08	72.91	Y	RN-50	ECCV 2020	Augustin2020Adversarial
14	Engstrom et al. [37]	90.83	69.24	N	RN-50	GitHub, Sep 2019	Engstrom2019Robustness
15	Rice et al. [113]	88.67	67.68	N	PreActRN-18	ICML 2020	Rice2020Overfitting
16	Rony et al. [114]	89.05	66.44	N	WRN-28-10	CVPR 2019	Rony2019Decoupling
17	Ding et al. [34]	88.02	66.09	N	WRN-28-4	ICLR 2020	Ding2020MMA
18	Standard	94.78	0.0	N	WRN-28-10	N/A	Standard

Table 5: Leaderboard for common corruptions, CIFAR-10.

	Model	Clean	Corr.	Extra data	Architecture	Venue	Model Zoo ID
1	Calian et al. [14]	94.93	92.17	Y	RN-50	arXiv, Apr 2021	N/A
2	Kireev et al. [73]	94.75	89.60	N	RN-18	arXiv, Mar 2021	Kireev2021Effectiveness_RLATAugMix
3	Hendrycks et al. [61]	95.83	89.09	N	ResNeXt29_32x4d	ICLR 2020	Hendrycks2020AugMix_ResNeXt
4	Hendrycks et al. [61]	95.08	88.82	N	WRN-40-2	ICLR 2020	Hendrycks2020AugMix_WRN
5	Kireev et al. [73]	94.77	88.53	N	PreActRN-18	arXiv, Mar 2021	Kireev2021Effectiveness_RLATAugMixNoJSD
6	Rebuffi et al. [111]	92.23	88.23	Y	WRN-70-16	arXiv, Mar 2021	Rebuffi2021Fixing_70_16_cutmix_extra_L2
7	Gowal et al. [50]	94.74	87.68	Y	WRN-70-16	arXiv, Oct 2020	N/A
8	Kireev et al. [73]	94.97	86.60	N	PreActRN-18	arXiv, Mar 2021	Kireev2021Effectiveness_AugMixNoJSD
9	Kireev et al. [73]	93.24	85.04	N	PreActRN-18	arXiv, Mar 2021	Kireev2021Effectiveness_Gauss50percent
10	Gowal et al. [50]	90.90	84.90	N	WRN-70-16	arXiv, Oct 2020	N/A
11	Kireev et al. [73]	93.10	84.10	N	PreActRN-18	arXiv, Mar 2021	Kireev2021Effectiveness_RLAT
12	Rebuffi et al. [111]	92.23	82.82	Y	WRN-70-16	arXiv, Mar 2021	Rebuffi2021Fixing_70_16_cutmix_extra_Linf
13	Gowal et al. [50]	91.10	81.84	Y	WRN-70-16	arXiv, Oct 2020	N/A
14	Gowal et al. [50]	85.29	76.37	N	WRN-70-16	arXiv, Oct 2020	N/A
15	Standard	94.78	73.46	N	WRN-28-10	N/A	Standard

Table 6: Leaderboard for the ℓ_∞-threat model, CIFAR-100.

	Model	Clean	Robust	Extra data	Architecture	Venue	Model Zoo ID
1	Gowal et al. [50]	69.15	36.88	Y	WRN-70-16	arXiv, Oct 2020	Gowal2020Uncovering_extra
2	Rebuffi et al. [111]	63.56	34.64	N	WRN-70-16	arXiv, Mar 2021	Rebuffi2021Fixing_70_16_cutmix_ddpm
3	Rebuffi et al. [111]	62.41	32.06	N	WRN-28-10	arXiv, Mar 2021	Rebuffi2021Fixing_28_10_cutmix_ddpm
4	Cui et al. [30]	62.55	30.20	N	WRN-34-20	ICCV 2021	Cui2020Learnable_34_20_LBGAT6
5	Gowal et al. [50]	60.86	30.03	N	WRN-70-16	arXiv, Oct 2020	Gowal2020Uncovering
6	Cui et al. [30]	60.64	29.33	N	WRN-34-10	ICCV 2021	Cui2020Learnable_34_10_LBGAT6
7	Rade and Moosavi-Dezfooli [107]	61.50	28.88	N	PreActRN-18	OpenReview, Jun 2021	Rade2021Helper_R18_ddpm
8	Wu et al. [155]	60.38	28.86	N	WRN-34-10	NeurIPS 2020	Wu2020Adversarial
9	Rebuffi et al. [111]	56.87	28.50	N	PreActRN-18	arXiv, Mar 2021	Rebuffi2021Fixing_R18_ddpm
10	Hendrycks et al. [60]	59.23	28.42	Y	WRN-28-10	ICML 2019	Hendrycks2019Using
11	Cui et al. [30]	70.25	27.16	N	WRN-34-10	ICCV 2021	Cui2020Learnable_34_10_LBGAT0
12	Chen et al. [22]	62.15	26.94	N	WRN-34-10	arXiv, Oct 2020	Chen2020Efficient
13	Sitawarin et al. [129]	62.82	24.57	N	WRN-34-10	ICML 2020	Sitawarin2020Improving
14	Rice et al. [113]	53.83	18.95	N	PreActRN-18	ICML 2020	Rice2020Overfitting

Table 7: Leaderboard for common corruptions, CIFAR-100.

	Model	Clean	Corr.	Extra data	Architecture	Venue	Model Zoo ID
1	Hendrycks et al. [61]	78.90	65.14	N	ResNeXt29_32x4d	ICLR 2020	Hendrycks2020AugMix_ResNeXt
2	Hendrycks et al. [61]	76.28	64.11	N	WRN-40-2	ICLR 2020	Hendrycks2020AugMix_WRN
3	Gowal et al. [50]	69.15	56.00	Y	WRN-70-16	arXiv, Oct 2020	Gowal2020Uncovering_extra_Linf
4	Gowal et al. [50]	60.86	49.46	N	WRN-70-16	arXiv, Oct 2020	Gowal2020Uncovering_Linf

Table 8: Leaderboard for the ℓ_∞-threat model, ImageNet.

	Model	Clean	Robust	Extra data	Architecture	Venue	Model Zoo ID
1	Salman et al. [117]	68.46	38.14	N	WRN-50-2	NeurIPS 2020	Salman2020Do_50_2
2	Salman et al. [117]	64.02	34.96	N	RN-50	NeurIPS 2020	Salman2020Do_R50
3	Engstrom et al. [37]	62.56	29.22	N	RN-50	GitHub, Oct 2019	Engstrom2019Robustness
4	Wong et al. [153]	55.62	26.24	N	RN-18	ICLR 2020	Wong2020Fast
5	Salman et al. [117]	52.92	25.32	N	RN-50	NeurIPS 2020	Salman2020Do_R18
6	Standard_R50	76.52	0.0	N	RN-50	N/A	Standard_R50

Table 9: Leaderboard for common corruptions, ImageNet.

	Model	Clean	Corr.	Extra data	Architecture	Venue	Model Zoo ID
1	Hendrycks et al. [62]	76.88	51.61	N	RN-50	ICCV 2021	Hendrycks2020Many
2	Hendrycks et al. [61]	76.98	46.91	N	RN-50	ICLR 2020	Hendrycks2020AugMix
3	Geirhos et al. [44]	74.88	44.48	N	RN-50	ICLR 2019	Geirhos2018_SIN_IN
4	Geirhos et al. [44]	77.44	40.77	N	RN-50	ICLR 2019	Geirhos2018_SIN_IN_IN
5	Standard_R50	76.52	38.12	N	RN-50	N/A	Standard_R50
6	Geirhos et al. [44]	60.24	37.95	N	RN-50	ICLR 2019	Geirhos2018_SIN
7	Salman et al. [117]	68.46	34.60	N	WRN-50-2	NeurIPS 2020	Salman2020Do_50_2_Linf