Empirical Analysis of Upper Bounds for Robustness Distributions using Adversarial Attacks

Aaron Berger*[2][0009−0006−3322−5718], Nils Eberhardt*[2][0009−0003−3964−8584], Annelot W. Bosman[1][0009−0004−1050−5165], Henning Duwe[2][0009−0008−4361−1203], Holger H. Hoos[1,2,3][0000−0003−0629−0099], and Jan N. van Rijn[1][0000−0003−2898−2168]

[1] Leiden Institute of Advanced Computer Science, Leiden University, The Netherlands
{a.w.bosman, j.n.van.rijn}@liacs.leidenuniv.nl
[2] Chair for AI Methodology, RWTH Aachen University, Germany
{duwe, hh}@aim.rwth-aachen.de
{aaron.berger, nils.eberhardt}@rwth-aachen.de
[3] University of British Colombia, Canada

Abstract. This study examines the effectiveness of adversarial attacks in determining upper bounds for robustness distributions for neural networks. While complete neural network verification techniques can provide exact safety margins, their computational cost limits scalability. To address this, we evaluate multiple adversarial attack methods, including FGSM, PGD, AutoAttack and FAB, comparing them to a state-of-the-art verification technique, α, β-CROWN. Using the MNIST dataset, we demonstrate that adversarial attacks yield computationally efficient and tight upper bounds for robustness distributions. We assess the trade-offs between running time, accuracy and the quality of the bounds obtained through our approach. The results highlight complementarities between verification and attack methods: Attacks achieve near-optimal upper bounds at a significantly reduced computational cost. These findings open opportunities for large-scale robustness analysis while acknowledging limitations in safety guarantees inherent to the approximation techniques on which our approach is based.

Keywords: Neural Network Verification · Adversarial Robustness · Distribution Analysis · Adversarial Examples

1 Introduction

In recent years, despite the remarkable performance of neural networks in various tasks, their vulnerability to adversarial perturbations has led to concerns about the reliability and security of these models. Adversarial perturbations are

* These authors contributed equally to this work

small, imperceptible modifications to inputs that lead to misclassifications. This rising concern is reflected in the sizeable body of scientific work investigating different kinds of attacks [1, 10, 11], training neural networks to be robust against perturbations [23, 29, 34] and formal verification of different robustness properties [19, 21, 24]. A prominent example of such a property is *local robustness*, which determines for a given network and input whether the network is robust against input perturbations up to a certain magnitude. Recently, Bosman *et al.* [3] introduced robustness distributions as a versatile method for analysing the robustness of a network and for comparing different networks for the same classification problem.

Originally, robustness distributions represent the amount of perturbation that can be applied to individual inputs given to a neural network such that correct outputs remain unaffected. These maximum allowable perturbation sizes for individual inputs are called critical epsilons, short ε^*; they are computed using neural network verification techniques. Neural network verification methods are limited in the type and size of networks they can verify since, amongst other reasons, verification is computationally expensive.

As a cheap alternative to measuring robustness distributions, we propose using adversarial attack methods to determine an upper bound on the ε^*-values. Unlike complete verification, which provides lower bounds on ε^*-values and, therefore, guarantees exact safety margins at a high computational cost, adversarial attacks provide much faster results at the cost of completeness; i.e., if no adversarial example is found this does not guarantee that none exists. This makes adversarial attack methods suitable for providing relatively cheap upper bounds to the ε^*-values, but not for making statements about the general robustness of neural networks. As an additional benefit, several attack methods can adapt to different network architectures, making them versatile compared to verification techniques that are usually constrained to specific architectures.

In the following, we compare multiple state-of-the-art adversarial attack methods to obtain upper bounds for robustness distributions to the baseline of robustness distributions obtained from complete verification methods. We empirically show that these approximations provide upper bounds of high quality. We use a binary search algorithm to determine the ε^* value for a given network and input. We have made the code for reproducing our experiments and the results available in a public GitHub repository.[4] In summary, the contributions of our work are as follows:

- We investigate the monotonicity assumption for perturbation radii of adversarial examples on MNIST, showing that it holds in most cases with few exceptions for attacks where randomness is included, namely pgd_40_random and AutoAttack. Binary search works optimally only if the problem is monotonic; this means that if an attack method can find an adversarial perturbation within a certain magnitude, it should also be able to find it within any larger magnitude, and vice versa. Note that while neural network ver-

[4] See: `https://github.com/ADA-research/Robustness-Estimation-Experiments`

ification methods strictly adhere to this monotonicity assumption, it is not necessarily satisfied by adversarial attack-based methods.
– We show that on the MNIST dataset, the upper bounds of robustness distributions can be approximated with attack-based methods that find comparable results to a high-performance complete verifier.
– We compare computational cost and quality of robustness distributions computed using complete verification and attack-based methods.
– We use the Kolmogorov-Smirnov test to show that approximating the robustness distributions on a subset of the data results in a distribution similar to that for the entire test dataset.
– We show performance complementarity between different state-of-the-art adversarial attacks and verifiers in terms of running time and the size of the adversarial perturbations obtained from them.

2 Background

In this section, we cover the background of local robustness verification and robustness distributions, as well as different adversarial attacks and other methods that try to estimate the distance of a data point to the decision boundary of a neural network.

2.1 Foundations of neural network verification

Neural network verification. Formally, a neural network classifier can be described by a function f_θ in $\mathbb{R}^n \to \mathbb{R}^m$, where θ is the set of trained parameters for f_θ, n is the number of input variables, and m is the number of possible classes. We use $F_y(x)$ to denote the probability that the classifier assigns to input x belonging to class y.

Considering an input x_0 with correct label $\lambda(x_0)$ and a region (also called ε-ball) around x_0 defined by $G_{p,\varepsilon}(x_0) = \{x : ||x - x_0||_p \leq \varepsilon\}$, using some norm l_p, local robustness verification aims to formally verify whether there exists a perturbed input $x \in G_{p,\varepsilon}(x_0)$, such that the predicted label of x differs from the true label $\lambda(x)$. In this paper, we exclusively focus on the ℓ_∞-norm due to its broad usage in the literature.

This problem can be modelled in the framework of mixed integer linear programming (MILP) [27]. As a solution to the MILP, we either find an adversarial perturbation or proof that an adversarial perturbation cannot exist within a certain ε radius. This problem of neural network verification, however, is NP-complete [18] and therefore computationally expensive, mainly due to the non-linearity of neural networks [27]. This is why complete verification does not scale well to state-of-the-art networks or to large datasets [5, 19], even though it has been the starting point for much work on complete verification techniques. One of these complete verification methods is α, β-CROWN [28], which has won the neural network verification competition in 2021, 2022, 2023 and 2024 [5–7].

α, β-CROWN is configured to first run adversarial attacks that can often efficiently disprove robustness within the given epsilon radius. α, β-CROWN uses the CROWN [33] bounding method as an incomplete verifier and uses branch-and-bound [8] to make the verification complete.

Robustness distributions. In the literature, robustness is usually quantified as a network's accuracy on a certain set of input instances and a fixed amount of perturbation ε [21]. This metric, known as robust accuracy is highly sensitive to small changes of the chosen ε. Comparing networks for different ε values can provide widely different views on the robustness of these networks. One recently introduced method for analysing the robustness of neural networks is the concept of robustness distributions [3]; these are empirical distributions of a certain robustness measure, such as the critical ε, over a set of input data and give a complete picture of the adversarial robustness of a neural network. Bosman *et al.* [3] used a complete verifier to find empirical lower bounds on the critical ε for a given neural network using k-binary search.

2.2 Methods for finding adversarial perturbations

In this work, we empirically investigate how well different algorithmic methods for finding adversarial examples [2, 12, 13, 16, 23] can provide upper bounds for robustness distributions. Unlike complete verification approaches, these methods cannot guarantee that no adversarial perturbation for a certain ε radius exists if they do not find one, but they are significantly less computationally expensive since they often only use a pre-defined number of gradient passes through the network and often still find imperceptible adversarial perturbations [16, 23]. In the following, we describe the different methods we considered for creating the empirical robustness distributions.

Fast Gradient Sign Method (FGSM) [16] is a one-step method for generating adversarial examples that utilise the loss-function of a neural network. FGSM is mainly designed for adversarially retraining neural networks to make them resistant to attacks. For each input variable in the input image x_0, we find the direction that maximises the loss function and multiply it by a predefined ε. Goodfellow et al. [16] find that this simple method leads to many misclassifications for a wide variety of networks. Since it was first published, FGSM has given rise to a wide range of extensions, such as Projected Gradient Descent, but they all still rely on the basic method that we use in this work.

Projected Gradient Descent (PGD) [23] utilises the gradient information of a neural network to find adversarial examples. This method can be seen as a multi-step version of FGSM. PGD starts with a random initialisation within the ε-ball around the given image. In each step, gradient descent on the negative loss is performed with a small learning rate. After each step, the current value is projected onto the ε-ball. PGD is commonly used in adversarial training [23, 26, 32], which can often increase robustness against adversarial attacks. Notably,

a variant of PGD is also used in the α, β-CROWN verifier to try to disprove robustness before starting the actual verification procedure.

Fast Adaptive Boundary Attack (FAB) [12] aims to produce minimally distorted adversarial examples, exploiting the computationally efficient box-constrained projections of the problem of finding p^*. This means that the minimisation problem is simplified to only consider upper and lower bounds on each of the input variables in x_0. This version of the optimisation problem is combined with the approximated decision boundaries of the network and a bias in the algorithm to find the closest possible adversarial example. This algorithm has a targeted and an untargeted version, meaning that we can either specify which class we want the classifier to misclassify the instance as or not. We refer to these two versions as targeted and untargeted FAB attack.

Square Attack [2] is a black-box attack based on iterative random search. In each iteration, the algorithm randomly modifies some pixels of the given image that are adjacent to each other and form a square within the input x from the previous iteration (or, in the beginning, the original image). The changes are only made if the resulting loss is larger than in the last step; loss in this context is defined as $F_y(x) - max_{k \neq y} F_k(x)$, where y is the original label. The stopping criterion is met as soon as an adversarial example is found. Square Attack is limited in the type of adversarial examples that it can find, however, it is a simple algorithm that does not rely on gradients and sometimes outperforms white-box attacks [13].

AutoAttack [13] is an ensemble method that combines two parameter-free versions of PGD with Square Attack and targeted FAB. In the evaluation of AutoAttack with models from the literature [13], the ensemble attack achieved lower robust test accuracy scores than reported in the original papers in all but one of the tested cases. AutoAttack is currently the standard attack method for evaluating robustness without guarantees in the context of adversarial training [11].

2.3 Other related work

Liu et al. [22] used a similar binary search-based approach to calculate the critical ε as we do in this work and considered lower bounds using an incomplete verifier. Carlini et al. [9] analysed the differences between provably minimal adversarial perturbations found by a verifier and adversarial perturbations found by an adversarial attack for the ℓ_1- and ℓ_∞-norms. They found by verification that iterative attacks perform better than single-step attacks and adversarial training increases the sizes of the minimal adversarial perturbations. Weng et al. [30] introduced a lower bound on the minimal adversarial perturbations with local Lipschitz constants. They used extreme value theory to approximate the maximal gradient over the high-dimensional ε-ball and provide an estimate of the lower bound. Notably, this method does not depend on a specific attack and can be applied to large networks.

3 Upper bounds to robustness distributions

Following Bosman *et al.* [3], we define the critical ε value, ε^*, for a trained neural network f_θ and an input x_0 with true class $\lambda(x_0)$ as the largest perturbation size such that the perturbed input $x \in \{x : \|x - x_0\|_p < \varepsilon^*\}$ is classified correctly, while any perturbation radius exceeding ε^* is guaranteed to permit misclassifications. Finding the exact ε^* is computationally complex and often practically infeasible. A possible solution to this is to discretise the search space for finding a lower bound to the ε^*. Then a variant of binary search can be used to traverse the search space efficiently, e.g. k-binary search, where k verification queries are solved simultaneously [3]. Here, for simplicity we rely on standard binary search. Binary search with complete verification does not only lead to a lower bound of the ε^*-value, which we will refer to as $\tilde{\varepsilon^*}$ we also obtain an upper bound of the value. We use the notion p^* for the smallest ε value that leads to misclassification and refer to it as the minimum adversarial perturbation. For conciseness, in the following, we use $\tilde{p^*}$, the value we find as an upper bound for p^*, and $\tilde{\varepsilon^*}$ to refer to the respective values in the discretised search space.

To obtain the baseline robustness distributions, we calculate the strict lower and upper bounds on the critical ε-value for each input image using binary search with a complete verification method [3]. For this baseline, we verify that the network is robust for the given input to obtain the lower bound $\tilde{\varepsilon^*}$, and an upper bound $\tilde{p^*}$. During binary search with complete verification, we possibly encounter out-of-memory errors and timeouts; to account for this, we use a slightly adapted version of the binary search, as outlined in Algorithm 1 in Appendix C. The estimation is tight if there exists no ε-value in the discretised range we consider that lies between the lower and upper bounds.

For the adversarial attacks, we only calculate the upper bound $\tilde{p^*}$ with binary search. We assume that smaller/larger ε-values than the last investigated can be disregarded. If an attack finds an adversarial example for the current ε, then $\tilde{\varepsilon^*}$ must be smaller. Otherwise, we continue the search with a larger ε, though adversarial examples may still exist for smaller values. Mathematically, the binary search assumes that the robustness result is monotonic concerning the perturbation size. If the monotonicity assumption holds, binary search will find the $\tilde{\varepsilon^*}$ being the closest to the true ε^*. We assume that if no adversarial example is found for some ε, none will be found for smaller values. This assumption is analysed in Section 5.1.

4 Setup of experiments

Choice of attack methods and complete verifier. We compare various methods that perform adversarial attacks. We selected AutoAttack since it is ubiquitously used to evaluate the robustness of large networks in the literature [11]. Additionally, we included the standard attacks FGSM and PGD. Furthermore, we included the FAB attack, since it is designed to find small perturbations. Notably, we refrained from using the commonly known DeepFool [25] attack, since

the FAB attack can be seen as a substantially improved version of DeepFool. Finally, we utilised the complete verifier α, β-CROWN, since it is considered state-of-the-art [5–7] to determine a reasonable upper and lower bound for the ε^*-values.

Experiment design. We perform binary search on perturbation sizes from the discretised search space $\mathcal{S} \coloneqq \{i/255 \mid i \in \mathbb{N}\} \cap [0, 0.4]$ to find a lower bound $\tilde{\varepsilon^*}$ and an upper bound $\tilde{p^*}$. The search space has the step size of pixel values, and the upper end is at 0.4, since for bigger values, no ε^*-values have been observed on MNIST [4]. We used the VERONA[5] package to run our experiments and compute the robustness distributions. We use the α, β-CROWN [28] verifier for verification with a time-out of 600 seconds for each verification query, where a verification query is the combination of one input and a specific perturbation radius.

Attack parameters. For all attacks except FAB, we use the binary search method introduced above and outlined in Algorithm 1. We use PGD with and without random initialisation, with $k = 40$ iterations, and we set the step size to $(0.2 - \frac{1}{255})/k$. We use the standard parameter settings of AutoAttack [13]. For FAB, we use the random initialisation of 0.3 from the original paper [12]. Notably, FAB is designed to find small perturbations such that no binary search is needed. We use five restarts of the untargeted and targeted FAB version. In each restart of the targeted version, an FAB attack is executed once for all possible target classes, i.e., nine times on MNIST. We assigned the maximum value (0.4) to instances where none of the queries produced an adversarial attack.

α, β-CROWN and AutoAttack both use PGD variants, and we chose the default setting for both of them, i.e., 100 PGD steps. α, β-CROWN, however, uses 30 restarts, while AutoAttack uses no restarts. In addition, AutoAttack uses a targeted PGD version on a special loss for nine classes, such that there are 10 runs of PGD variants in total used within AutoAttack.

Choice of input images. In this work, we analysed the robustness distributions for different MNIST networks. We selected the same 100 testing images as in [3, 19] for the robustness distributions using complete verification. We discarded originally misclassified images for each network and did not include them in the distributions; the resulting number of images per network can be found in Appendix A, Table 3.

Choice of networks. We selected four fully connected networks with ReLU activation functions. These networks have been extensively used in the literature [3]. The `mnist_relu_m_n` networks have n layers with m units and were trained using standard methods. `mnist-net_256x2` has two layers with 256 units. More details about the model architectures are provided in Appendix A, Table 3.

[5] `https://github.com/ADA-research/VERONA`

Execution environment. All experiments were carried out on one NVIDIA H100 GPU with 96 GB of RAM. We executed the binary search for four images simultaneously on the same GPU using multiple threads for the α, β-CROWN verifier.

Performance metrics. We computed four performance metrics to compare the methods we investigated: the *average running time*, the *average minimal adversarial perturbation sizes* $\bar{p^*}$, the *ratio to the best* $\bar{p^*}$ (RB-$\bar{p^*}$) and the *relative marginal contribution* (RMC). RB-$\bar{p^*}$ denotes the ratio of $\bar{p^*}$ found by the given algorithm divided by $\bar{p^*}$ of the *virtual best algorithm* (VBA), which is inspired by the virtual best solver concept from Xu et al. [31]. The VBA for a given input image is the method that found the smallest adversarial perturbation. If more than one algorithm found the smallest adversarial perturbation, we choose the method with the lowest running time as the VBA. We follow the work of König et al. [19] for the definition of marginal contribution. The RMC is the fraction of inputs on which a given method found a smaller adversarial perturbation than all others; thus, the RMC quantifies the impact of adding or removing a given algorithm from the ensemble containing all of them.

5 Results

In the following, we report the results of our empirical analysis. First, we investigate the underlying assumption for using binary search to compute robustness values. Second, we show the adversarial upper bounds compared to the robustness distribution obtained by using the exact verifier. Third, we compare the upper bounds per instance to assess the quality of the bounds we have obtained. Then, we report the running time needed for our different methods and investigate the marginal contribution of the various methods. Last, we show the adversarial upper bounds of the robustness distributions computed using the entire testing set.

5.1 Monotonicity assumption of perturbation radii of adversarial examples

To investigate the monotonicity assumption on the MNIST dataset for each of the adversarial attack methods, we iteratively checked the local robustness for each ε-value in our search space for all networks. We computed the distance between the minimum adversarial example $\tilde{p^*}$ and the largest $\tilde{\varepsilon^*}$ for which we found the network to be robust. If we have a robust result for an ε bigger than our $\tilde{p^*}$, we have a counterexample to the monotonicity assumption. As can be seen in Appendix B, Table 4, the monotonicity assumption does not hold for all input images. While we could not find counterexamples to the monotonicity for FGSM and PGD, we did encounter counterexamples for both adversarial attack methods where randomness was used for creating the perturbed image, namely *pgd_40_random* and AutoAttack.

For AutoAttack, we see that the monotonicity assumption does not hold for one query. For PGD with random initialisation, four counterexamples to the monotonicity assumption were found for each network. Interestingly, those were found to involve the same images across the four networks, showing that the perturbation sizes for these adversarial examples are similar across the different networks. Out of the 1572 queries to find the $\tilde{\varepsilon^*}$ with an iterative search procedure, the monotonicity assumption did not hold for just 17 queries. For all of these, the distances to the optimal value were no more than 5 steps in our discretised search space. As these counterexamples to the monotonicity assumption occur rarely, we argue that using binary search is valid for the computation of the robustness values. However, it should be noted that this will not always result in optimal upper bounds compared to iterative search.

5.2 Comparison of complete verification and adversarial attacks

Next, we analysed whether adversarial attack methods can provide a tight upper bound on the minimum adversarial example. We approach this by first comparing the adversarial attack methods to the minimum adversarial examples computed using our baseline verifier α, β-CROWN per image and then by looking at the resulting robustness distributions from the different algorithms. We visualise the robustness distributions in Figure 1. Most of the adversarial attack methods lead to $\tilde{p^*}$-values close to the ones obtained using complete verification, and the computed $\tilde{p^*}$-values are close to the $\tilde{\varepsilon^*}$-values for most of the networks. In contrast to this, for the biggest network, `mnist_relu_4_1024`, the gap between $\tilde{p^*}$ and $\tilde{\varepsilon^*}$-values is more significant. One can see that especially the minimum adversarial perturbation values from α, β-CROWN and AutoAttack are very similar. Also, for most of the networks, PGD and FAB attacks lead to similar distributions. In general, the worst approximations are computed using FGSM, which shows the biggest gap to the other distributions.

By testing the hypothesis of log-normality using the Kolmogorov-Smirnov (K-S) test [14] with a confidence level of 0.05, we found evidence that most of the robustness distributions constructed using the adversarial attack methods we considered tend to follow log-normal distributions. This is in line with findings recently reported in the literature [3]. Only for the `mnist−net_256x2` with the pgd_40 attack, the null hypothesis of log-normality is rejected, see Appendix F, Table 8.

In Figure 2, we compare AutoAttack with α, β-CROWN on a per-instance level, showing that while most $\tilde{p^*}$-values seem to be the same, there exist some complementarities between the complete verifier and adversarial attack methods.

Table 1 contains the performance results from our experiments. Using the adversarial attack methods, we observed running times two to four magnitudes smaller than those for complete verification. At the same time, adversarial attack methods do not result in the same robustness distributions as when the distributions are computed using complete verification. When relying on adversarial attack methods, we compute upper bounds on robustness and do not obtain formal safety guarantees. Still, adversarial attack methods can produce

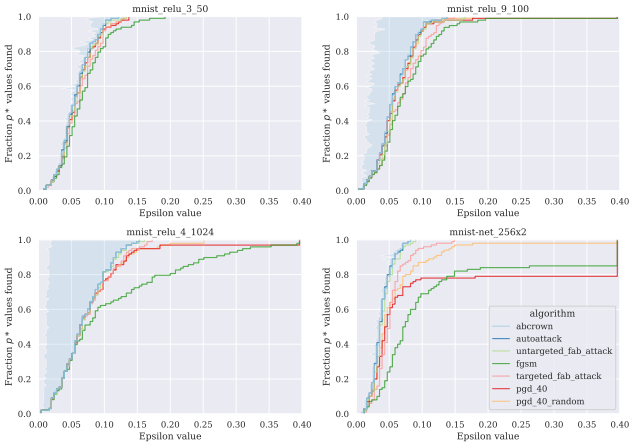


Fig. 1: Approximation techniques and complete verification on first 100 MNIST test images. The lines display the CDFs of the $\tilde{p}^*$-values for the different methods. The lighter blue area indicates the gap between $\tilde{\varepsilon}^*$ and $\tilde{p}^*$ values from α, β-CROWN.

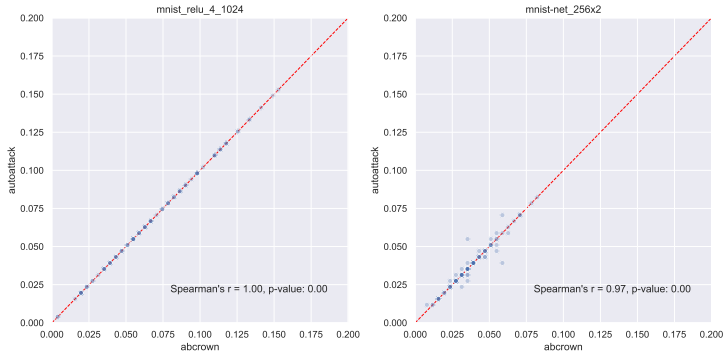


Fig. 2: Scatterplots of the minimum adversarial examples found with α, β-CROWN (x-axis) and AutoAttack (y-axis) for two networks. Each point in the figure represents one image from the MNIST test set.

upper bounds of robustness distributions with reasonable quality and significantly smaller running time for the MNIST dataset.

We find that on the 100 images subset, α, β-CROWN is the best-performing method with an RMC of 0.066 and a $\bar{p}^*$-value of 0.005. All the RMCs are quite low, because α, β-CROWN and AutoAttack obtain the same $\tilde{p}^*$ on 89% of the instances, as shown in Appendix F Figure 4. AutoAttack found a smaller $\tilde{p}^*$ for 4.3% of the instances and for 6.3% of the instances α, β-CROWN found a smaller $\tilde{p}^*$, as listed in Appendix F Figure 5 for all pairs of methods. Figure 5

	100 images				Complete test set			
algorithm	Time [s]	$\bar{p^*}$	RMC	RB-$\bar{p^*}$	Time [s]	$\bar{p^*}$	RMC	RB-$\bar{p^*}$
VBA	290.787	0.055	-	1.000	7.316	0.063	-	1.000
abcrown	3731.513	0.055	0.066	1.005	-	-	-	-
autoattack	20.233	0.055	0.028	1.006	19.999	0.063	0.214	1.002
fgsm	0.047	0.095	0.000	1.741	0.046	0.107	0.000	1.695
pgd_40	0.139	0.080	0.000	1.471	0.140	0.089	0.002	1.407
pgd_40_random	0.143	0.064	0.000	1.167	0.142	0.075	0.000	1.184
targeted_fab	0.333	0.064	0.000	1.164	0.451	0.074	0.000	1.171
untargeted_fab	0.550	0.058	0.003	1.052	0.555	0.069	0.003	1.092

Table 1: Performance comparison of complete verification method α, β-CROWN and the considered adversarial attack methods in terms of running time in seconds averaged per image, the average minimum adversarial example, the relative marginal contribution (RMC) and the ratio of the average minimum adversarial perturbation $\bar{p^*}$ over compared to the virtual best algorithm (VBA). Whenever multiple methods find the same $\bar{p^*}$, we consider the method with the smallest running time to be part of the VBA.

shows that only for 0.69% of the instances any of the other methods was able to obtain a smaller $\tilde{p^*}$ than AutoAttack or α, β-CROWN.

On 158 out of our 1572 test instances, we find that for the robustness values computed using the verifier, $\tilde{p^*} - \tilde{\varepsilon^*} = 1/255$, and thus we find the optimal approximation for these values in our discretised search space. In Table 2, we compare the $\tilde{p^*}$-values of the various methods for these instances. We note that in 90.5% of the cases, AutoAttack finds the same $\tilde{\varepsilon^*}$ as α, β-CROWN, and for the cases where AutoAttack did not find the optimal $\tilde{\varepsilon^*}$, it obtained a value just 0.9% larger than the $\tilde{\varepsilon^*}$ from α, β-CROWN. Regarding the other attacks, there is a significant performance gap, as they find the optimal values in less than 70% of the cases. For the test images under consideration, the average running time for the verifier is significantly smaller than for all the instances, which could indicate that the instances where we find tight approximations are also instances where it is easier to compute the local robustness properties.

5.3 Comparison of attacks on the complete test set

As we have shown, using adversarial attack methods we can efficiently compute good upper bounds for robustness distributions. As the current research on robustness distributions [3, 4] is limited to small data subsets, due to the computational complexity of complete verification, we see the use of attack-based methods as a means to computing robustness distributions on larger datasets. In Figure 3, we compare the robustness distributions computed using the adversarial attack methods on the complete MNIST test set with 10000 images to the robustness distributions presented in the previous section. While most of the

algorithm	time [s]	$\bar{p^*}$	f_{opt}	RB-$\bar{p^*}$
VBA	11.330	0.040	1.000	1.000
abcrown	82.993	0.040	1.000	1.000
autoattack	17.312	0.040	0.905	1.009
fgsm	0.042	0.083	0.146	2.083
pgd_40	0.115	0.072	0.551	1.824
pgd_40_random	0.116	0.046	0.557	1.157
targeted_fab_attack	0.314	0.050	0.316	1.247
untargeted_fab_attack	0.488	0.042	0.677	1.051

Table 2: Statistics on the 158 test images for which the ε^* computation of α, β-CROWN is as tight as possible and it holds that $\tilde{p^*} - \tilde{\varepsilon^*} = 1/255$. f_{opt} denotes the frequency of finding the size of the smallest possible adversarial perturbation.

distributions seem similar for the two datasets, for `mnist-net_256x2` a long tail appears on the full test set that has been absent on the first 100 images. This shows either that the first 100 images give a too conservative view of the robustness of this network or we find worse quality upper bounds for this network on the whole dataset due to an unidentified factor. In contrast to the distributions on the first 100 images, we could not find evidence for the distributions on the entire test set to follow log-normal distributions using K-S tests. This is expected, as the K-S test tends to become overly conservative for large samples of discrete data, for a comprehensive overview see the work of Darling [14]. Similar to the results on the first 100 images, AutoAttack finds the tightest upper bounds on the complete test set, followed closely by the FAB attacks and PGD with random initialisation. FGSM and PGD without random initialisation were found to obtain the loosest bounds.

As we do not have $\tilde{\varepsilon^*}$-values from complete verification for the whole test dataset, we cannot use the distance between the $\tilde{\varepsilon^*}$-values and the minimum adversarial perturbations values as a measure of the quality of the distributions. Instead, we investigate if the distributions on the whole test set are statistically similar to the ones from the subset; which can be used as an indicator of the quality of the distributions. Using the K-S test we compare the distributions on the subset and the whole test set in Appendix F Table 10. We find general similarities between the results on the whole test dataset and the subset robustness distributions, indicating that looking at a subset of the MNIST dataset should generally be sufficient for computing a robustness distribution and showing that we can also approximate robustness distributions on a whole test set with reasonable quality.

While we observed many similarities between the robustness distributions on the first 100 test images, that does not hold for the complete test set, as we show in Appendix 7. There, for all except one combination, using pairwise K-S tests, we did not find evidence that the distributions are similar. This is possibly due to the rank of performances we found on the 100 images, where

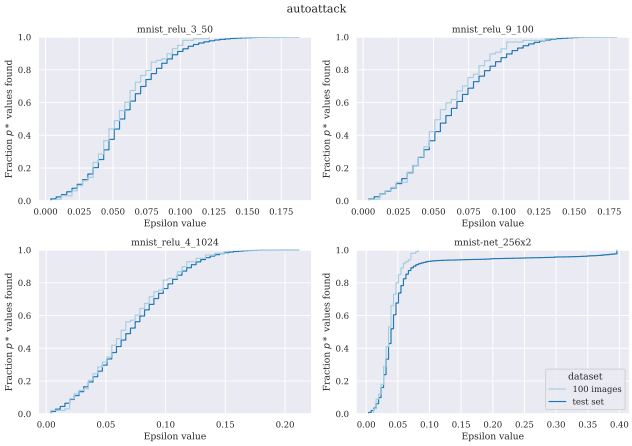


Fig. 3: To assess the similarities and differences of the robustness distributions on the 100-image subset and the test set, we plot the respective CDFs. Here we show the results for AutoAttack and all networks used in our study.

AutoAttack performs significantly better than the other attacks. We assume that these performance differences between the algorithms get magnified on the whole test dataset, which leads to statistically different distributions.

6 Discussion and conclusions

In this work, we investigated the use of approximation methods based on adversarial attacks for determining high-quality upper bounds for robustness distributions. We have shown that adversarial attack methods can provide tight upper bounds at significantly lower computational costs than required for running complete verification methods. These upper bounds can be used to obtain a relatively cheap estimation of the robustness of a given neural network. Furthermore, we examined the feasibility of using binary search and provided evidence that for adversarial attacks with non-random initialisation, our search space is mostly monotonic.

Our results pave the way to estimating robustness distributions for larger datasets. Building on existing work from the literature, we observed similarities between log-normal distributions and our upper-bound robustness distributions on a small subset of the MNIST dataset. We also find no significant difference in the distributions on the subset and that on the complete dataset, while our statistical tests suggest that the distributions on the entire dataset do not follow log-normal distributions; this is likely due to our test becoming overly conservative for large samples of discrete data. Because of this, the similarity of the robustness distributions on the whole dataset cannot be guaranteed, as complete verification of the whole dataset was not feasible.

As the approach introduced here is more scalable than the earlier use of complete verification techniques, it would be interesting to apply it to more complex datasets, such as CIFAR-10 [20], GTSRB [17] and ImageNet [15], to consider more sophisticated network architectures and to further investigate the quality of the approximations thus obtained for adversarially and certifiably trained neural networks.

Overall, we believe that to use robustness distributions in practical applications, we need to determine good lower and upper bounds. Therefore, future research on methods for finding tight and inexpensive lower bounds for robustness distributions is needed.

Acknowledgements

The authors thank Konstantin Kaulen for his helpful feedback on this paper. This research was partially supported by TAILOR, a project funded by EU Horizon 2020 research and innovation program under GA No. 952215, and by an Alexander-von-Humboldt Professorship in AI held by Holger Hoos. Computations were performed using computing resources granted by RWTH Aachen University under project rwth1650.

References

1. Akhtar, N., Mian, A.: Threat of adversarial attacks on deep learning in computer vision: A survey. IEEE Access **6**, 14410–14430 (2018)
2. Andriushchenko, M., Croce, F., Flammarion, N., Hein, M.: Square attack: a query-efficient black-box adversarial attack via random search. In: European Conference on Computer Vision. pp. 484–501 (2020)
3. Bosman, A.W., Hoos, H.H., van Rijn, J.N.: A preliminary study of critical robustness distributions in neural network verification. In: Workshop on Formal Methods for ML-Enabled Autonomous Systems (FOMLAS) (2023)
4. Bosman, A.W., Münz, A.L., Hoos, H.H., van Rijn, J.N.: A preliminary study to examining per-class performance bias via robustness distributions. In: The 7th International Symposium on AI Verification (SAIV). pp. 116–133 (2024)
5. Brix, C., Bak, S., Johnson, T.T., Wu, H.: The fifth international verification of neural networks competition (VNN-COMP 2024): Summary and results. arXiv preprint arXiv:2412.19985 (2024)
6. Brix, C., Bak, S., Liu, C., Johnson, T.T.: The fourth international verification of neural networks competition (VNN-COMP 2023). arXiv preprint arXiv:2312.16760 (2023)
7. Brix, C., Müller, M.N., Bak, S., Johnson, T.T., Liu, C.: First three years of the international verification of neural networks competition (VNN-COMP). International Journal on Software Tools for Technology Transfer **25**(3), 329–339 (2023)
8. Bunel, R., Lu, J., Turkaslan, I., Torr, P.H., Kohli, P., Kumar, M.P.: Branch and bound for piecewise linear neural network verification. Journal of Machine Learning Research (JMLR) **21**(42), 1–39 (2020)
9. Carlini, N., Katz, G., Barrett, C., Dill, D.L.: Provably minimally-distorted adversarial examples. arXiv preprint arXiv:1709.10207 pp. 1–8 (2018)

10. Chakraborty, A., Alam, M., Dey, V., Chattopadhyay, A., Mukhopadhyay, D.: A survey on adversarial attacks and defences. CAAI Transactions on Intelligent Technology **6**(1), 25–45 (2021)
11. Costa, J.C., Roxo, T., Proença, H., Inácio, P.R.M.: How deep learning sees the world: A survey on adversarial attacks & defenses. IEEE Access **12**, 61113–61136 (2024)
12. Croce, F., Hein, M.: Minimally distorted adversarial examples with a fast adaptive boundary attack. In: International Conference on Machine Learning (ICML). pp. 2196–2205 (2020)
13. Croce, F., Hein, M.: Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In: International Conference on Machine Learning (ICML). pp. 2206–2216 (2020)
14. Darling, D.A.: The Kolmogorov-Smirnov, Cramér-von Mises Tests. The Annals of Mathematical Statistics **28(4)**, 823–838 (1957)
15. Deng, J., Dong, W., Socher, R., Li, L., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 248–255 (2009)
16. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. In: International Conference on Learning Representations (ICLR). vol. 3, pp. 1–10 (2015)
17. Houben, S., Stallkamp, J., Salmen, J., Schlipsing, M., Igel, C.: Detection of traffic signs in real-world images: The German Traffic Sign Detection Benchmark. In: International Joint Conference on Neural Networks. pp. 1–8 (2013)
18. Katz, G., Barrett, C., Dill, D.L., Julian, K., Kochenderfer, M.J.: Reluplex: An efficient smt solver for verifying deep neural networks. In: International Conference on Computer Aided Verification (CAV). pp. 97–117. Springer (2017)
19. König, M., Bosman, A.W., Hoos, H.H., van Rijn, J.N.: Critically assessing the state of the art in neural network verification. Journal of Machine Learning Research (JMLR) **25**(12), 1–53 (2024)
20. Krizhevsky, A., Nair, V., Hinton, G.: Cifar-10 (Canadian institute for advanced research)
21. Li, L., Xie, T., Li, B.: Sok: Certified robustness for deep neural networks. In: IEEE symposium on Security and Privacy. pp. 1289–1310 (2023)
22. Liu, J., Chen, L., Miné, A., Yu, H., Wang, J.: Input validation for neural networks via local robustness verification. In: International Conference on Software Quality, Reliability, and Security Companion (QRS-C). pp. 237–246 (2023)
23. Madry, A., Makelov, A., Schmidt, L., Tsipras, D., Vladu, A.: Towards deep learning models resistant to adversarial attacks. In: International Conference on Learning Representations (ICLR). pp. 1–15 (2018)
24. Meng, M.H., Bai, G., Teo, S.G., Hou, Z., Xiao, Y., Lin, Y., Dong, J.S.: Adversarial robustness of deep neural networks: A survey from a formal verification perspective. IEEE Transactions on Dependable and Secure Computing pp. 1–18 (2022)
25. Moosavi-Dezfooli, S.M., Fawzi, A., Frossard, P.: Deepfool: a simple and accurate method to fool deep neural networks. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 2574–2582 (2016)
26. Singh, N.D., Croce, F., Hein, M.: Revisiting adversarial training for ImageNet: Architectures, training and generalization across threat models. In: Advances in Neural Information Processing Systems (NeurIPS). vol. 36 (2023)
27. Tjeng, V., Xiao, K.Y., Tedrake, R.: Evaluating robustness of neural networks with mixed integer programming. In: International Conference on Learning Representations (ICLR). pp. 1–11 (2019)

28. Wang, S., Zhang, H., Xu, K., Lin, X., Jana, S., Hsieh, C.J., Kolter, J.Z.: Beta-CROWN: Efficient bound propagation with per-neuron split constraints for complete and incomplete neural network verification. In: Advances in Neural Information Processing Systems (NeurIPS). vol. 34 (2021)
29. Wang, Y., Ma, X., Bailey, J., Yi, J., Zhou, B., Gu, Q.: On the convergence and robustness of adversarial training. In: International Conference on Machine Learning (ICML). pp. 6586–6595 (2019)
30. Weng, T.W., Zhang, H., Chen, P.Y., Yi, J., Su, D., Gao, Y., Hsieh, C.J., Daniel, L.: Evaluating the robustness of neural networks: An extreme value theory approach. In: International Conference on Learning Representations (ICLR). pp. 1–14 (2018)
31. Xu, L., Hutter, F., Hoos, H., Leyton-Brown, K.: Evaluating component solver contributions to portfolio-based algorithm selectors. In: International Conference on Theory and Applications of Satisfiability Testing (SAT). pp. 228–241 (2012)
32. Zhang, H., Yu, Y., Jiao, J., Xing, E.P., Ghaoui, L.E., Jordan, M.I.: Theoretically principled trade-off between robustness and accuracy. In: International Conference on Machine Learning (ICML). pp. 7472–7482 (2019)
33. Zhang, H., Weng, T., Chen, P., Hsieh, C., Daniel, L.: Efficient neural network robustness certification with general activation functions. In: Advances in Neural Information Processing Systems (NeurIPS). vol. 31, pp. 4944–4953 (2018)
34. Zhao, W., Alwidian, S., Mahmoud, Q.H.: Adversarial Training Methods for Deep Learning: A Systematic Review. Algorithms **15**(8), 283 (2022)

A Model information

network	neurons	parameters	linear layers	subset instances	test instances
`mnist_relu_3_50`	0.1K	42K	3	98	9588
`mnist−net_256x2`	0.5K	269K	3	100	9808
`mnist_relu_9_100`	0.8K	150K	9	97	9508
`mnist_relu_4_1024`	3K	2913K	4	98	9385

Table 3: Architecture details about the used networks from König et al. [19] with the number of correctly classified instances on the 100 images subset and the MNIST test set.

B Monotonicity Experiment

attack	network	number images	step distance mean
autoattack	`mnist_relu_3_50`	1	1
pgd_40_random	`mnist−net_256x2`	5	3
	`mnist_relu_3_50`	5	3
	`mnist_relu_4_1024`	5	3
	`mnist_relu_9_100`	5	3

Table 4: Results of the monotonicity experiment. The experiment was executed on the first 100 images of the MNIST test set. We show for each algorithm-verifier combination, for how many images the monotonicity assumption was violated (where we found the network to be safe for a specific ε bigger than p^*). To show how the bounds could have been looser by using binary search, we include the mean step distance from p^* (with our step size $1/255$) to the highest ε with a safe result. Algorithm-verifier combinations with no counterexample are omitted.

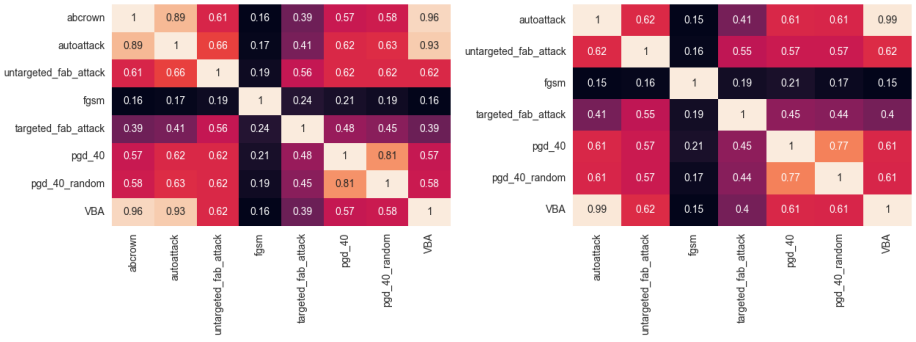
C Binary Search with α, β-crown

Algorithm 1: Binary search algorithm

Data: Ordered set of epsilon values E with $|E| = n$, a verifier $f(x, \varepsilon \in E) = y \in \{safe, unsafe, error\}$, an image x
Result: critical epsilon ε^*, minimum adversarial perturbation p^*

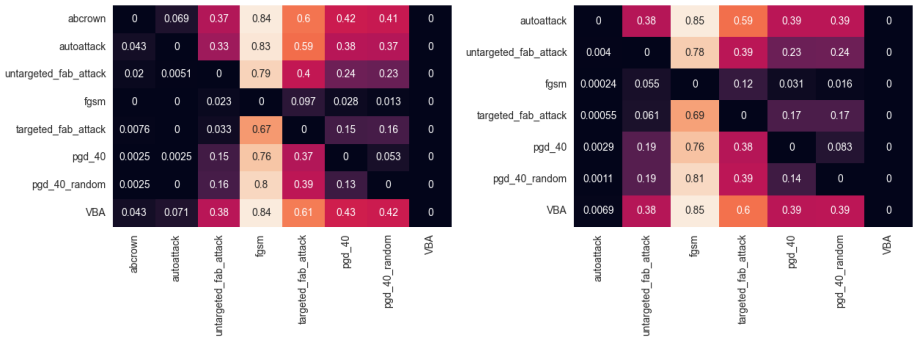
$first = 0$;
$last = n - 1$;
$results = \{\}$;
while $first \leq last$ **do**
  $midpoint = (first + last)//2$;
  $\varepsilon = E[midpoint]$;
  $result = f(x, \varepsilon)$;
  $results = results \cup \{(result, \varepsilon)\}$;
  if $result == unsafe$ **then**
    $last = midpoint - 1$;
  else if $result == safe$ **then**
    $first = midpoint + 1$;
  else
    $E = E \setminus \{\varepsilon\}$;
    $last = last - 1$
$\varepsilon^* = getBiggestSafeEpsilon(results)$;
$p^* = getSmallestUnsafeEpsilon(results)$;
return (ε^*, p^*)

D Pairwise comparison of algorithms



(a) First 100 images (b) Complete test set

Fig. 4: Fraction of instances where two algorithms found a minimal adverarial perturbation of equal size.



(a) First 100 images (b) Complete test set

Fig. 5: Fraction of instances where the method on th y-axis found a smaller minimal adverarial perturbation than the method on the x-axis.

E Running time and minimal adversarial perturbations statistics

	Time				p^*			
algorithm	mean	min	max	std	mean	min	max	std
VBA	7.32	0.03	45.63	11.08	0.063	0.004	0.396	0.047
autoattack	20.00	6.49	45.63	7.82	0.063	0.004	0.396	0.047
fgsm	0.05	0.03	6.30	0.03	0.107	0.004	0.396	0.095
pgd_40	0.14	0.09	0.97	0.03	0.089	0.004	0.396	0.095
pgd_40_random	0.14	0.09	0.80	0.04	0.075	0.004	0.396	0.066
targeted_fab	0.45	0.19	9.16	1.06	0.074	0.004	0.400	0.055
untargeted_fab	0.56	0.38	5.80	0.21	0.069	0.004	0.400	0.057

Table 5: Execution time in seconds and minimal adversarial perturbation on the complete MNIST testing set. We do not investigate α, β-CROWN for the entire testing set, as this is computationally infeasible.

	Time				p^*			
algorithm	mean	min	max	std	mean	min	max	std
VBA	290.79	0.04	24,716.12	2,031.69	0.055	0.004	0.153	0.028
abcrown	3,731.51	31.17	24,716.12	4,732.15	0.055	0.004	0.153	0.028
autoattack	20.23	6.60	45.63	8.25	0.055	0.004	0.153	0.028
fgsm	0.05	0.03	1.40	0.07	0.095	0.004	0.396	0.087
pgd_40	0.14	0.09	0.46	0.04	0.080	0.004	0.396	0.088
pgd_40_random	0.14	0.09	0.27	0.04	0.064	0.004	0.396	0.044
targeted_fab	0.33	0.28	1.70	0.10	0.064	0.004	0.173	0.033
untargeted_fab	0.55	0.42	1.26	0.13	0.058	0.004	0.165	0.030

Table 6: Execution time in seconds and minimal adversarial perturbation on the first 100 MNIST test images.

F Statistical analysis

network	algorithm	measure	statistic	p-value	passed
`mnist_relu_3_50`	abcrown	p^*	0.06	0.81	True
		ε^*	0.07	0.64	True
`mnist_relu_9_100`	abcrown	p^*	0.07	0.78	True
		ε^*	0.52	0.00	False
`mnist_relu_4_1024`	abcrown	p^*	0.06	0.89	True
		ε^*	0.53	0.00	False
`mnist-net_256x2`	abcrown	p^*	0.09	0.42	True
		ε^*	0.08	0.48	True

Table 7: K-S test for log-normality for the robustness distributions computed of different neural networks using α, β-CROWN on the first 100 images for the ε^* and the p^* values. The tests with a p-value of less than 0.05 were rejected. We provide the test statistic and the p-value as well as the interpretation of the test, i.e., whether the null-hypothesis was rejected (False) or accepted (True).

network	algorithm	statistic	p-value	passed
mnist_relu_3_50	*abcrown*	0.06	0.81	True
	autoattack	0.06	0.88	True
	untargeted_fab_attack	0.07	0.71	True
	fgsm	0.05	0.97	True
	targeted_fab_attack	0.06	0.85	True
	pgd_40	0.06	0.82	True
	pgd_40_random	0.05	0.91	True
mnist_relu_9_100	*abcrown*	0.07	0.78	True
	autoattack	0.07	0.76	True
	untargeted_fab_attack	0.08	0.48	True
	fgsm	0.07	0.76	True
	targeted_fab_attack	0.09	0.38	True
	pgd_40	0.07	0.65	True
	pgd_40_random	0.06	0.79	True
mnist_relu_4_1024	*abcrown*	0.06	0.89	True
	autoattack	0.06	0.89	True
	untargeted_fab_attack	0.06	0.88	True
	fgsm	0.07	0.73	True
	targeted_fab_attack	0.05	0.94	True
	pgd_40	0.06	0.88	True
	pgd_40_random	0.05	0.95	True
mnist-net_256x2	*abcrown*	0.09	0.42	True
	autoattack	0.08	0.55	True
	untargeted_fab_attack	0.09	0.36	True
	fgsm	0.11	0.16	True
	targeted_fab_attack	0.09	0.33	True
	pgd_40	0.16	0.01	False
	pgd_40_random	0.10	0.30	True

Table 8: K-S test for log-normality for the robustness distributions computed using attacks on the first 100 images. The tests with a p-value of less than 0.05 were rejected.

network	algorithm	statistic	p-value	passed
mnist_relu_3_50	*autoattack*	0.04	0.00	False
	untargeted_fab_attack	0.04	0.00	False
	fgsm	0.03	0.00	False
	targeted_fab_attack	0.03	0.00	False
	pgd_40	0.04	0.00	False
	pgd_40_random	0.04	0.00	False
mnist_relu_9_100	*autoattack*	0.04	0.00	False
	untargeted_fab_attack	0.03	0.00	False
	fgsm	0.04	0.00	False
	targeted_fab_attack	0.04	0.00	False
	pgd_40	0.06	0.00	False
	pgd_40_random	0.03	0.00	False
mnist_relu_4_1024	*autoattack*	0.04	0.00	False
	untargeted_fab_attack	0.04	0.00	False
	fgsm	0.06	0.00	False
	targeted_fab_attack	0.04	0.00	False
	pgd_40	0.05	0.00	False
	pgd_40_random	0.03	0.00	False
mnist-net_256x2	*autoattack*	0.14	0.00	False
	untargeted_fab_attack	0.17	0.00	False
	fgsm	0.14	0.00	False
	targeted_fab_attack	0.16	0.00	False
	pgd_40	0.18	0.00	False
	pgd_40_random	0.11	0.00	False

Table 9: K-S test for log-normality for the robustness distributions computed using attacks on all MNIST test images. The tests with a p-value of less than 0.05 were rejected.

algorithm	network	p-value	passed
autoattack	mnist_relu_3_50	0.69	True
	mnist-net_256x2	0.03	False
	mnist_relu_9_100	0.51	True
	mnist_relu_4_1024	0.67	True
fgsm	mnist-net_256x2	0.31	True
	mnist_relu_3_50	0.10	True
	mnist_relu_9_100	0.27	True
	mnist_relu_4_1024	0.43	True
pgd_40	mnist_relu_9_100	0.41	True
	mnist-net_256x2	0.18	True
	mnist_relu_3_50	0.67	True
	mnist_relu_4_1024	0.69	True
pgd_40_random	mnist_relu_4_1024	0.52	True
	mnist_relu_9_100	0.32	True
	mnist_relu_3_50	0.47	True
	mnist-net_256x2	0.07	True
targeted_fab_attack	mnist_relu_9_100	0.48	True
	mnist_relu_4_1024	0.62	True
	mnist_relu_3_50	0.52	True
	mnist-net_256x2	0.02	False
untargeted_fab_attack	mnist_relu_3_50	0.54	True
	mnist_relu_9_100	0.14	True
	mnist-net_256x2	0.01	False
	mnist_relu_4_1024	0.80	True

Table 10: K-S test to compare the minimum adversarial perturbation distributions for the first 100 images with the distributions for the whole dataset. The tests with a p-value of less than 0.05 were rejected.

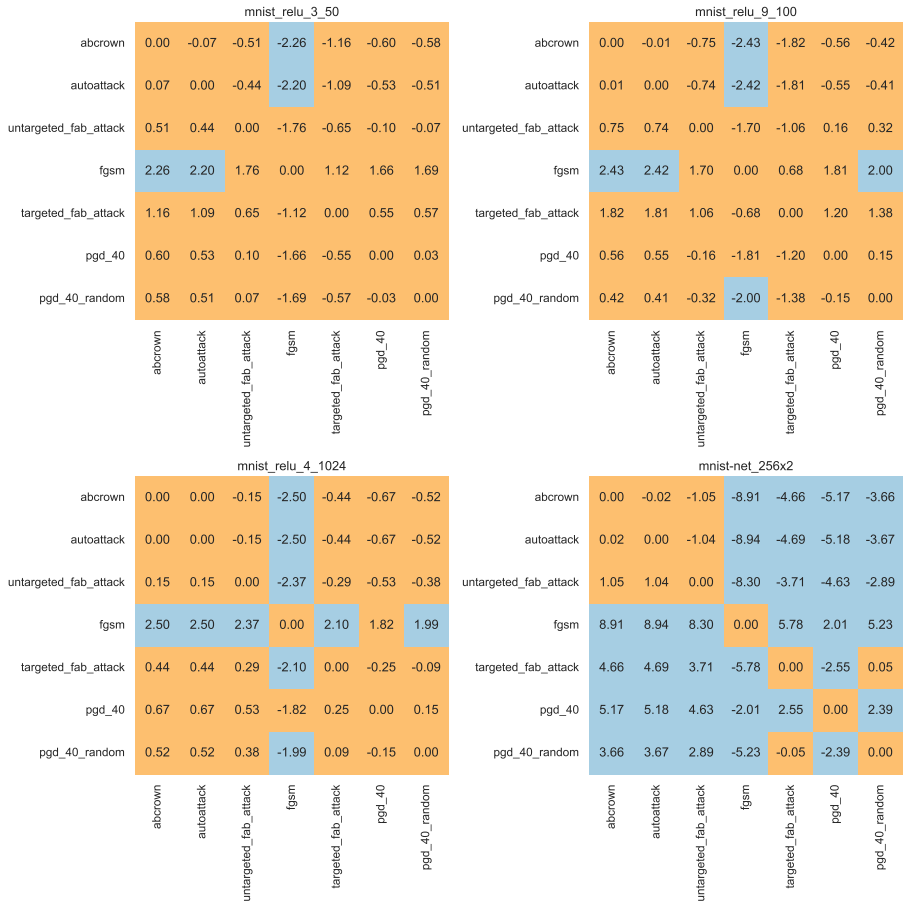


Fig. 6: Pairwise T-tests on the first 100 images. The plot shows the test statistics for the different combinations. The combinations that passed the test are highlighted in yellow.

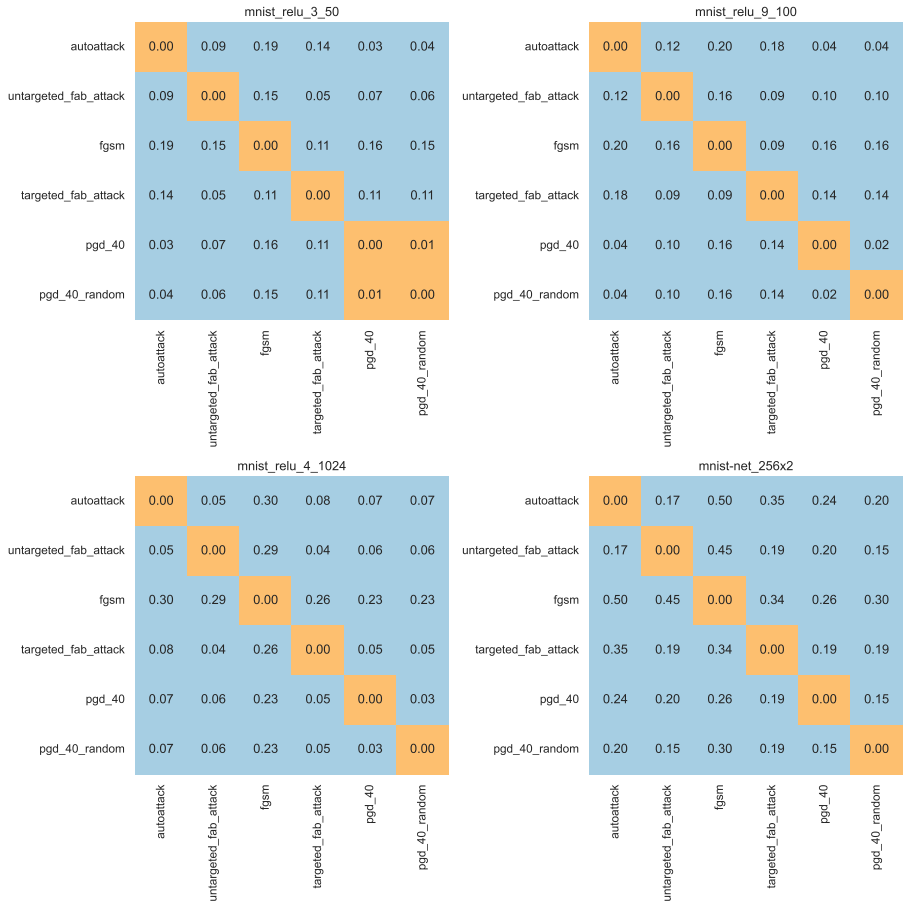


Fig. 7: Pairwise K-S tests on the whole MNIST test set. The plot shows the test statistics for the different combinations. The combinations that passed the test are highlighted in yellow.

G Visual Analysis

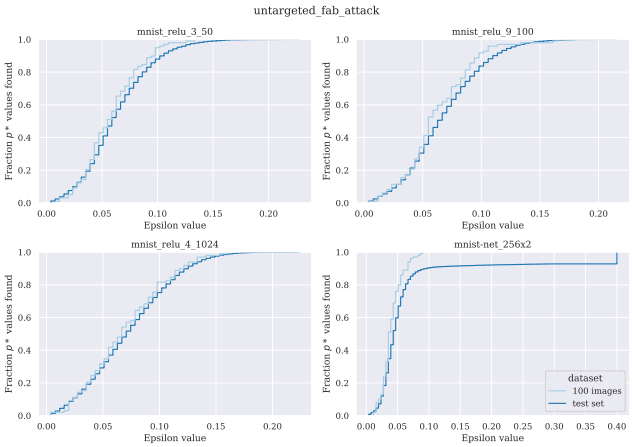


Fig. 8: To assess the similarities and differences of the robustness distributions on the 100 images subset and the test set, we plot the cdf functions of both distributions together. Here we show the results for the $untargeted\_fab\_attack$ attack and all the networks we used.

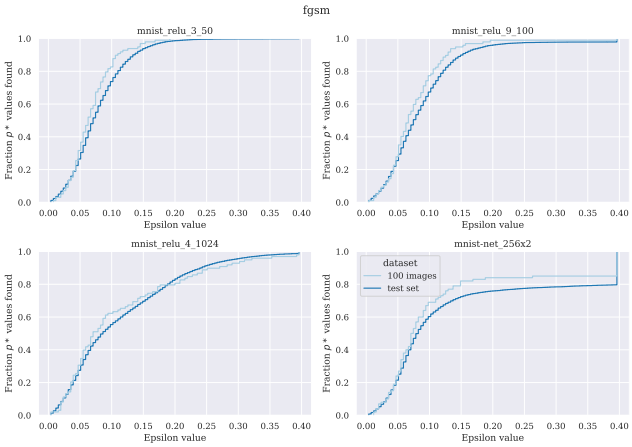


Fig. 9: To assess the similarities and differences of the robustness distributions on the 100 images subset and the test set, we plot the cdf functions of both distributions together. Here we show the results for the $fgsm$ attack and all the networks we used.

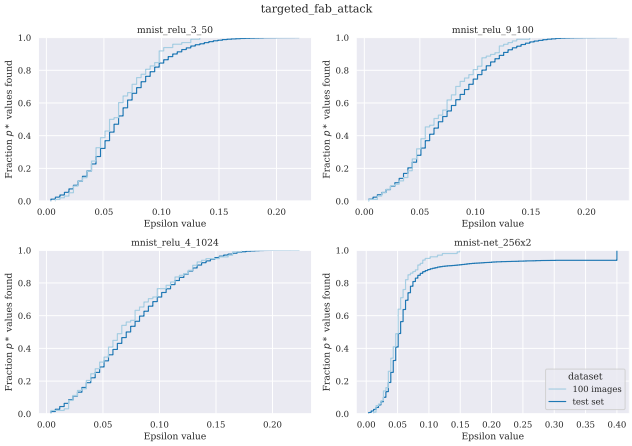


Fig. 10: To assess the similarities and differences of the robustness distributions on the 100 images subset and the test set, we plot the cdf functions of both distributions together. Here we show the results for the $targeted\_fab\_attack$ attack and all the networks we used.

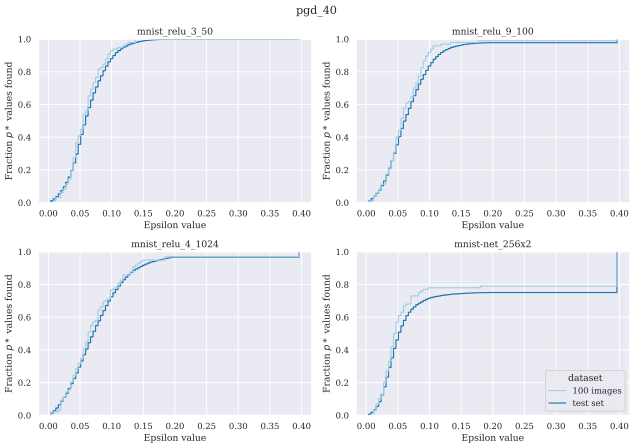


Fig. 11: To assess the similarities and differences of the robustness distributions on the 100 images subset and the test set, we plot the cdf functions of both distributions together. Here we show the results for the pgd_40 attack and all the networks we used.

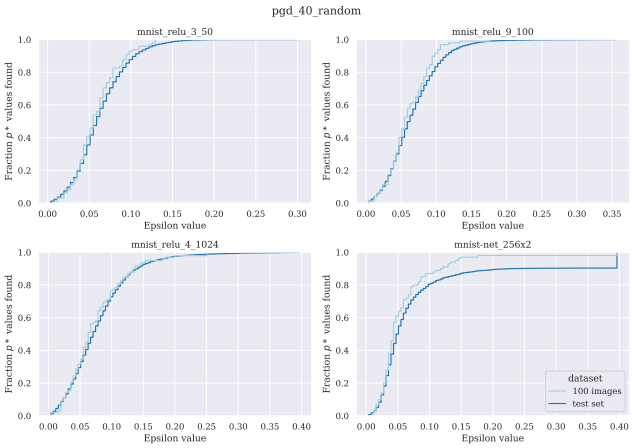


Fig. 12: To assess the similarities and differences of the robustness distributions on the 100 images subset and the test set, we plot the cdf functions of both distributions together. Here we show the results for the *pgd_40_random* attack and all the networks we used.