A Preliminary Study of Critical Robustness Distributions in Neural Network Verification

Annelot W. Bosman[1], Holger H. Hoos[1,2,3], and Jan N. van Rijn[1]

[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
hh@cs.rwth-aachen.de
[3] University of Britisch Colombia
Canada

Abstract. Neural networks are vulnerable to slight alterations to correctly classified inputs, leading to incorrect predictions. To rigorously assess the robustness of neural networks against such perturbations, exact verification techniques are employed. Robustness is generally measured in terms of adversarial accuracy, based on an upper bound on the magnitude of perturbations commonly denoted as epsilon. For each input in a given set, it is then determined whether a perturbation up to magnitude epsilon can deceive the network. In this work, we demonstrate that by refining the notion of a single bound epsilon as well as the analysis of neural network robustness, interesting insights can be gained. We introduce the concept of critical epsilon values, defined as the maximum amount of perturbation for which a given input is provably correctly classified, such that any larger perturbations can cause misclassification. To effectively estimate the critical epsilon values for each input in a given set, we utilise a variant of the binary search algorithm that we call k-binary search. We then analyse the distribution of critical epsilon values over a given set of inputs for 12 classifiers that have been used widely in the literature on neural network verification. Using a Kolmogorov-Smirnov test, we found support for the hypothesis that the critical epsilon values of all of the networks follow a log-normal distribution. We also analyse the distribution of epsilons over a set of previously unseen inputs (testing data) and the Kolmogorov-Smirnov test finds no statistically significant difference compared to seen inputs (training data). Interestingly, we find that input that is easily corrupted to deceive one network may require a considerably larger perturbation to deceive another.

Keywords: neural network verification · adversarial robustness · binary search algorithm · distribution analysis

1 Introduction

It is well known that neural networks are vulnerable to producing incorrect outputs resulting from changes in inputs. One commonly researched type of vulnerability is adversarial attacks, based on small input perturbations. Various research [10, 19] has shown how small, sometimes imperceptible changes to an input can mislead even state-of-the-art neural networks. Therefore, robustness against small input perturbations, also known as local robustness, is a crucial aspect to investigate.

Various methods for assuring complete robustness against perturbations within a certain radius have since been developed [1–3, 6–9, 11, 12, 20–22, 24]. Yearly neural network verification competitions have been held since 2020, aiming to critically asses the performance of different tools available and to incentivise their broad use by means of standardised formats [16]. Verification tools have increased in speed as well as scalability, in part by utilising algorithms from the field of optimisation and GPU acceleration. There exist two types of neural network verification algorithms: (i) complete algorithms and (ii) incomplete algorithms. In contrast to incomplete algorithms, complete verification methods, if run to completion, provide formal proof of whether or not a neural network is robust with respect to a certain input [14]. This comes at the cost of additional computational complexity.

In many works, the size of the input perturbation radius, epsilon, is seen as a parameter that is set in advance [14]. Following this notion, *robust accuracy* is defined as the percentage of inputs that will be classified correctly, regardless of what perturbation within the predefined bound epsilon will be added to it [25]. Thus, current state-of-the-art tools give for each instance a binary answer to the question of whether a certain perturbation could lead to misclassification.

However, it is our conviction that robust accuracy does not fully describe the robustness of a network, for the following reasons: (i) it requires the domain expert to pre-specify an acceptable perturbation level, which is knowledge that might not be available at that point; (ii) it does not indicate what level of perturbation is tolerated for an individual input before it will be misclassified; and (iii) when comparing two networks with similar robustness accuracy, several important nuances might not be revealed with respect to the robustness observed for individual inputs. For example, when for one network many of the inputs tolerate only a perturbation slightly larger than the pre-defined epsilon value, and for another neural network this margin is much larger, this is information that should be useful to anyone interested in the robust use of the network.

To overcome these limitations, in this work, we are interested in the behaviour of the robustness distributions over input instances of neural networks. In order to study this, we define the concept of *critical epsilon value*, which is defined as the maximum amount of perturbation to a given input for which the predicted class will provably remain correct. We use a complete verifier to provide the robustness guarantee. In this work, we will utilise a method from the literature, k-binary search [4], to empirically estimate a lower-bound on the critical epsilon value. By running multiple verification queries in parallel, we can make use of the

information obtained from the query that finishes first. In particular, this helps us to determine whether we need to continue running the other verification queries, and/or produce a number of new queries. While Liu *et al.* [15] have aimed to do the same with a traditional binary search algorithm, they were not able to do so for the computationally challenging case of complete formal verification methods, but rather performed this on the case of incomplete verification methods, resulting in an overapproximation of the actual critical epsilon value for many instances. As such, to the best of our knowledge, our work is the first to analyse the robustness distributions of complete verification methods.

The set of critical epsilon values behaves like a distribution rather than a threshold, which we call the *robustness distribution* of a given network. An advantage of using distributions to define the robustness of a network is that we can make assumptions about inputs for which we have not found the critical epsilon values.

The main contributions of our work presented here are as follows:

1. We introduce the concept of critical epsilon values, which are defined as the maximum amount of perturbation that provably does not lead to a misclassification for a given input is, such that any larger perturbations could result in a misclassification.
2. We utilise a variant of the binary search algorithm, k-binary search, in order to identify the critical epsilon values for a given network. When searching for the critical epsilon for a given network and input, we encounter uncertain delays and possibly missing values. k-binary search handles this by evaluating k epsilon values in parallel, potentially terminating queries early when new information becomes available.
3. We analyse the behaviour of robustness distributions of 12 widely used neural networks on both training data and testing data. Surprisingly, we observe that the robustness distributions of the neural networks we analysed follow log-normal distributions. Furthermore, robustness distributions do not significantly differ between training and testing data, suggesting that these distributions generalise to previously unseen inputs.

2 Background

In this section, we cover the background on local robustness verification, as well as the verification method we utilise, and the work of Liu *et al.* [15], who have looked at robustness distributions in a different context.

Local robustness verification: Verification methods for neural networks aim to formally evaluate whether a given model satisfies certain input-output properties. One such property is local robustness, which refers to the ability of a neural network to maintain correct predictions even when inputs are slightly perturbed. To assess local robustness, verification algorithms analyse a set of inputs and seek to determine whether there exist small perturbations that could cause the model to produce incorrect outputs. Such small perturbations added to the formally correctly classified image is called an *adversarial example*. The size

of these perturbations is restricted by a predefined maximum radius, typically denoted by ε, which limits the extent of changes that can be made to each input variable.

Formally, a deep 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 trainable parameters for f_θ, n is the number of input variables and m is the number of possible classes.

Assume an input x_0 with correct label $\lambda(x_0)$ and a region around x_0 defined by $G_{p,\varepsilon}(x_0) = \{x : ||x - x_0||_p < \varepsilon\}$. Local robustness verification aims to formally answer whether there exists a perturbed input $x \in G_{p,\varepsilon}(x_0)$ such that the predicted label of x is no longer equal to the predicted label of x_0. In this research, the perturbation is measured by the l_∞ norm.

Local robustness verification is typically modelled using mixed-integer program (MIP) formulations, which is an NP-complete problem. If an adversarial example exists, it will be found if the solver is given enough time. However, as neural networks can have many millions of variables, current verifiers do not scale well to state-of-the-art networks. Verification tools have increased in speed as well as scalability, in part by utilising algorithms from the field of optimisation [2, 11, 12] as well as GPU acceleration [6, 18, 22]. There also exist incomplete verification tools; however, these do not provide the guarantee of an answer, even if run arbitrarily long.

Branch-and-Bound-based neural network verification: In our experiments we utilise a recent version of the Branch-and-Bound-based neural network verification framework (BaB) [3, 6]. BaB tackles the verification problem by converting it into a global mixed integer programming (MIP) formulation and subsequently solving it using the branch-and-bound algorithm. The branch-and-bound algorithm partitions the feasible region of the MIP formulation into smaller regions, making it easier to solve the MIP iteratively.

Robustness distributions for incomplete verification methods: Setting out to validate inputs, Liu *et al.* [15] introduce the use of the binary search algorithm to find what we call the critical epsilon. Their study utilised the CLEVER method [23], which estimates minimal distortion, as well as ERAN with RefineZono [18] and DeepZono [17], which are a complete and incomplete verification method, respectively. Their work was limited to an analysis of the distribution of critical epsilon for one small network using complete verification, citing the costliness of the process for the other neural networks they studied. Additionally, they assumed that the distribution of robustness follows a normal distribution, which stands in contradiction to our results presented later.

3 Robustness distributions

In this section, we present the definition of critical epsilons as well as k-binary search [5], the algorithm we utilise to find critical epsilon values. These two concepts are necessary to obtain the robustness distributions we are investigating in this work.

3.1 Critical epsilon values

We define the critical epsilon value ε^*, for a trained neural network f_θ and input x_0 with correct class $\lambda(x_0)$, as the maximum perturbation size such that any perturbed input $x \in \{x : ||x - x_0||_p < \varepsilon^*\}$ cannot lead to misclassification and any perturbation larger that ε^* will provably lead to misclassification.

Existing complete verifiers are able to verify whether an adversarial example exists within a certain radius ε and provide the adversarial example if one is found. However, these methods are currently not able to determine the aforementioned critical epsilon ε^*.

Therefore we need to find alternative methods to accomplish this. Verifying every possible ε is infeasible, because verification can take considerable time, due to the computational complexity of the problem. In order to utilise the advantages of (binary) search methods, we will discretise the problem and investigate a predefined set of epsilon ranges.

In this research, we define $\tilde{\varepsilon}^*$ as the empirical lower-bound to ε^*. Assume we investigate a set of epsilon values $\mathcal{E} = \{\varepsilon_0, ..., \varepsilon_n\}$ for a given network f_θ and input x_0, where $\varepsilon_i < \varepsilon_{i+1}, \forall i \in \{0, \ldots, n-1\}$. If there exists an adverserial example for ε_i, where $0 < i \leq n$, and no adverserial example for ε_{i-1}, $\tilde{\varepsilon}^*$ is determined at ε_{i-1}. The exact critical epsilon, ε^* will be in the range $[\varepsilon_{i-1}, \varepsilon_i)$. In the case where perturbation with maximum value ε_i does not lead to misclassification and ε_{i+j} does lead to a misclassification, where $0 \leq i < n$ and $0 < j <= n - i$ and the verifiers are not able to resolve the verification queries for epsilon values in between these two in the set (e.g., due to time-outs or memory issues), $\tilde{\varepsilon}^*$ will be determined at ε_i as a conservative lower-bound.

3.2 *k*-binary search

A naïve approach to finding the critical epsilons is to verify all values in $\mathcal{E} = [\varepsilon_0, ..., \varepsilon_n]$. However, this is inefficient in terms of resource use, as solving a single verification query is already compute-intensive.

For each verification query, the verification of whether or not a specific epsilon value with a specific network and input leads to an adversarial example takes an unknown amount of time, and we define a time-out for each query. This means that a binary search approach experiences delays and potentially contains missing results, if the time-out is reached. If we use a regular binary search, as done by Liu *et al.* [15], it is possible that the benefits of binary search are nullified by the delays and missing values.

Instead, we propose k-binary search, a modified version of binary search, based on the work of Cicalese *et al.* [4]. The key idea is to verify k queries in parallel, rather than just one at a time. In their work, Cicalese *et al.* [4] analysed the optimal strategy for binary search when there are unknown delays and at most one missing value. In k-binary search, we apply this strategy to verify multiple queries of the same network and input with different epsilon values at once. Normal binary search typically selects the midpoint of a sorted array, checks the answer and iteratively continues until the correct element is found.

When verifying multiple queries in parallel, we divide the search space into $k+1$ parts. In this research we choose to use equal-sized parts. Using information from the queries that run in parallel but find a result faster, we can therefore reduce the total time necessary to find the critical epsilon values.

For example, suppose we are simultaneously verifying query A and B with epsilons of value a and b respectively, where $a > b$ for a given input and network. If A terminates before B, in some cases we can immediately use the result of A to infer the result of B. Specifically, if for a, no adversarial example is found, we know that there will be no adversarial example found for b, and we can terminate that process early. This allows us to save computing resources by avoiding unnecessary verification runs.

Using the k-binary search on the discretised problem will not provide us with the exact critical epsilon; instead, the algorithm gives us a range within which the exact critical epsilon lies. For example, if we find an adversarial example for an epsilon of size 0.005, but there does not exist one for 0.003, we take 0.003 as our empirical bound for the critical epsilon value, knowing that the actual critical epsilon lies in the range $(0.003, 0.005]$. In case we find, for example, for epsilon 0.003 there exists no adversarial example and then for one or more consecutive epsilons we find errors or time-outs and for the next epsilon, we find an adversarial example we again determine 0.003 as the lower-bound to the critical epsilon. We have thus discretised the problem of finding an estimation of the critical epsilon for a given input.

4 Empirical investigation

In our computational experiments, we pursued the following goals. Firstly, we explore whether k-binary search is able to determine reasonable lower-bounds on the critical epsilon values. Secondly, we investigate whether the critical robustness distributions follow the same parametric distribution class for all neural networks. Finally, we assess whether the critical robustness of the training observations and the testing observations come from the same distribution for each network.

4.1 Experimental setup

Network selection: For this experiment, 12 pre-trained MNIST neural networks were used. These neural networks were also part of the work by König *et al.* [13] and are generally widely used in the literature on neural network verification. We only consider neural networks that were not adversarially trained, in order to see how the robustness distributions of standard neural networks behave. An investigation of the behaviour of robustness distributions for adversarially trained neural networks is left for future work. While the aforementioned work investigates 15 neural networks, three of these lead to various errors, caused by input incosistency and out-of-memory issues, which is why we omitted these from our study.

Our research is limited to examining MNIST networks that have been extensively studied in the literature on neural network verification. Although some CIFAR classifiers are commonly used in such studies, their accuracy is generally low. For this reason, we chose not to include CIFAR networks in our research, as we believe the robustness distributions of classifiers with such low accuracy may not accurately represent the behaviour of state-of-the-art neural networks.

Instance selection: In this work, we use the first 100 MNIST training-set images and 100 MNIST testing-set images. We only use an image for a specific network if the image was originally correctly classified, meaning that the set of images in the distribution might vary over networks. The reason for this is that the neural networks do not originally misclassify the same inputs. In total, we attempted to find the critical epsilons for 1 147 verification queries with training images and 1 154 verification queries with testing images. Each verification query used a maximum of 3 CPU cores and 20GB of memory. The time-out for each sub-verification query was set to one hour.

Algoritm setup: We conducted a k-binary search with a range of epsilon values from 0.001 to 0.4, in intervals of 0.002, resulting in a length of 40 for the epsilon range. Cicalese *et al.* [4] analyse the exact maximum length n of values in the range for k-binary search such that there exists a winning of t queries when at most k queries are started simultaneously. Using their formula, we find that using two simultaneous queries will result in a theoretical maximum of 11 queries to find the critical epsilon within the 40 intervals. However, we note that Cicalese *et al.* [4] assume a maximum of one time-out in the epsilon range. In our setting, we have an unknown number of time-outs, which renders the problem more complex. Using two parallel queries, we split the remaining search space into equal-sized parts every time. When one of the queries finishes yet the other query is not fished and we have no new information about the unfinished query, we will find a new ε that is in the midpoint of the new range of epsilons.

Hardware: The experiments were carried out on a cluster of machines equipped with Intel Xeon E5-2683 CPUs with 32 cores, 40 MB cache size and 94 GB RAM, running CentOS Linux 7.

4.2 Critical robustness distributions on training data

Figure 1 shows a boxplot of the robustness distribution of training set of the 12 MNIST classifiers. It shows that for different epsilon values, different networks would be preferable based on robust accuracy.

For example, suppose that we had chosen an epsilon value of 0.012, which is used widely in practice, network *relu_4_1024* has a robust accuracy of 93% and network *net* a better robust accuracy of 100%. However, had we chosen an epsilon value of 0.04, network *relu_4_1024* and *net* have a robust accuracy of 66% and 49% respectively. While this is not reflected by the minimal critical epsilon, network *relu_4_1024* has the highest median and could be seen as one of the most robust networks over the entire set. In Figure 1, we can see two reference lines at 0.012 and 0.04 to further illustrate this example.

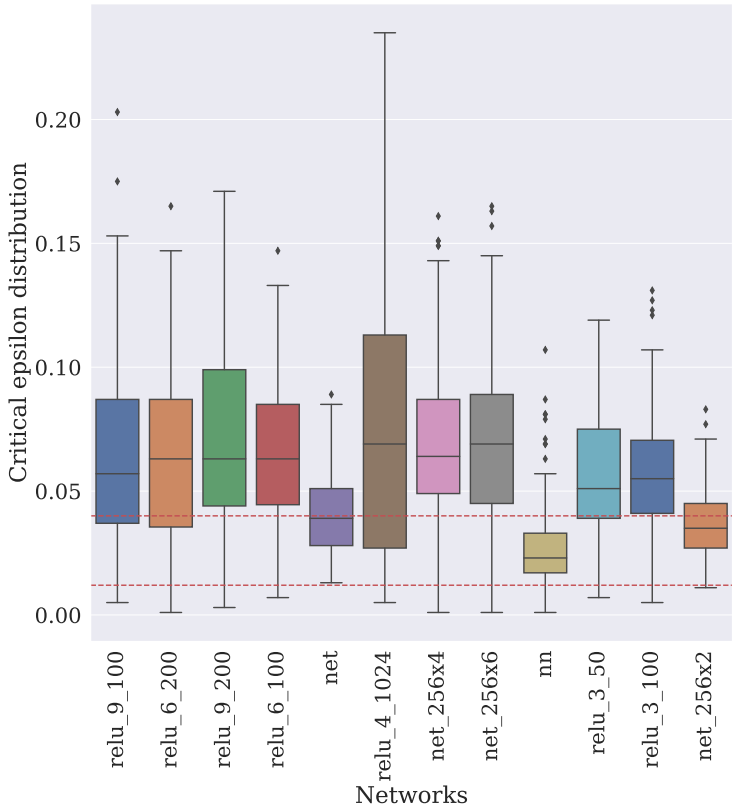
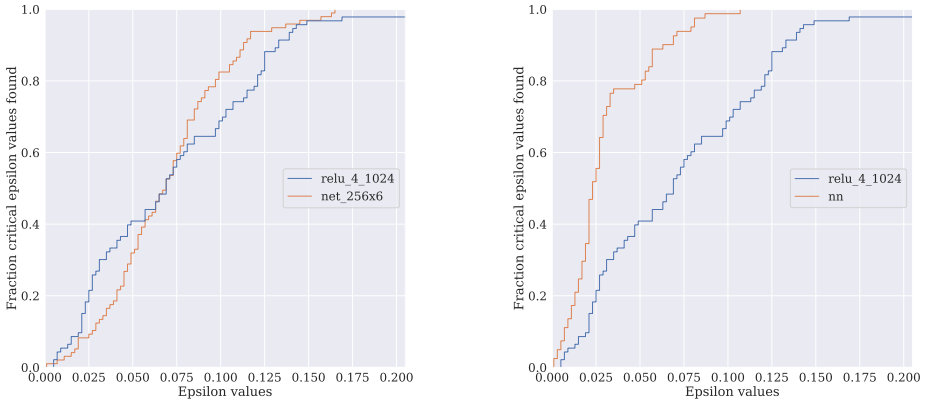


Fig. 1: Boxplot of the distribution of $\tilde{\varepsilon}^*$ for 12 MNIST classifiers over their set of correctly classified inputs out of the training data considered.

Figures 2a and 2b display the cumulative distribution functions of the robustness distribution for the classifier with the highest median and the classifier with the second-highest median and lowest median, respectively. In Figure 2b it can be seen that the robustness distribution of network *relu_4_1024* is better than that of network *nn*. However, in Figure 2a, we cannot make such a claim. It is evident that, for approximately the 50% least robust images, the *net_256x6* network has a higher $\tilde{\varepsilon}^*$ values.

We use the Kolmogorov-Smirnov test with a standard confidence level of 0.05 to investigate whether the robustness distributions of the training set are significantly different from a log-normal distribution. For none of the networks we find evidence that their robustness distribution of the training data is significantly different from a log-normal distribution.

Finding that the critical epsilon values follow a specific distribution means that the critical epsilon values of data we have not assessed is likely to follow the



(a) (b)

Fig. 2: Cumulative distribution of the fraction of instances for which $\tilde{\varepsilon}^*$ has been found for the training data, for three different networks for comparison.

same distribution. This implies that it is possible to determine the robustness of a neural network without the need to evaluate every conceivable input.

4.3 Critical robustness distributions on testing data

Figure 3 contains the boxplots of the robustness distributions for the approximate critical epsilons, $\tilde{\varepsilon}^*$, for the training set and testing sets. Clearly, the boxplots of each network's testing distribution is fairly similar to that of the corresponding training distributions.

Figures 4a and 4b show the CDF of the training and testing data for two different networks. We show in Figure 4a the network with the highest median and in 4b the CDF of the neural network with the lowest median of the networks considered. Empirically, we can see that the distributions of the training and testing data are very similar.

We use the Kolmogorov-Smirnov test with a standard confidence level of 0.05 to investigate whether the robustness distributions of the testing set are significantly different for each of the networks. For none of the networks, we find evidence that their robustness distribution of the testing data is significantly different from the distribution of the training data. This could imply that finding the robustness distributions for the training set is sufficient for analysing the overall robustness of a network in a supervised learning scenario. However, at this point, we have limited empirical evidence for this claim, and more research is needed to confirm it further.

4.4 Behaviour of k-binary search

We analyse several aspects of the behaviour of k-binary search.

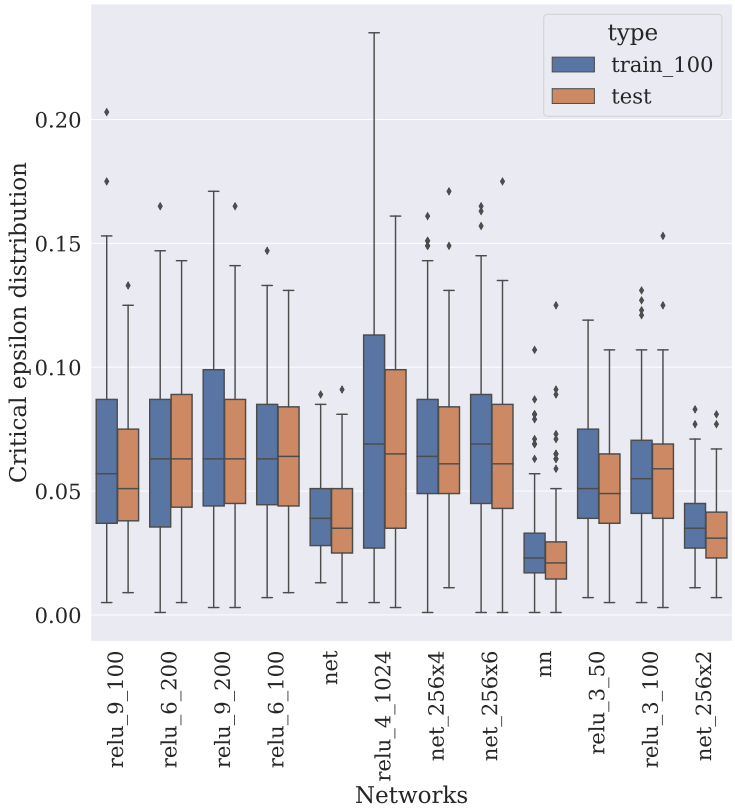
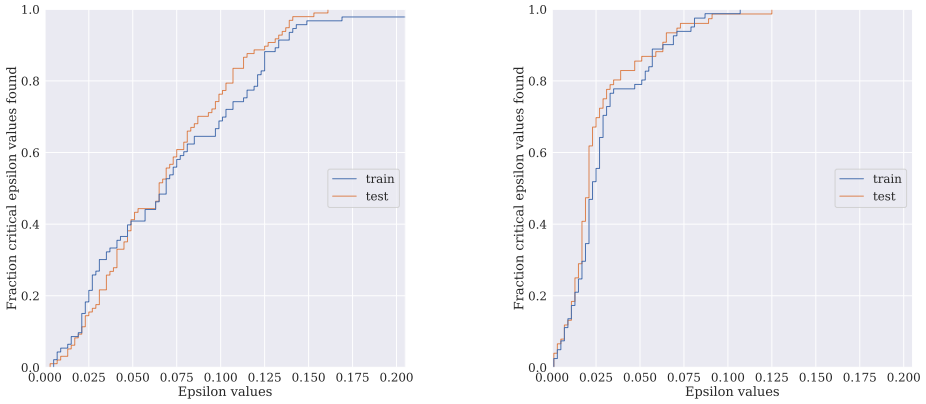


Fig. 3: Boxplot of the distribution of $\tilde{\varepsilon}^*$ for 12 MNIST classifiers over their set of correctly classified inputs out of the training data and testing data considered.

Critical terminations: One consequence of using verification queries with a time-out is that we might encounter time-outs of critical epsilons. This means that we get one or more time-outs for epsilons in between two epsilons for which the smallest does not lead to an adversarial example and the larger does. However, examining our dataset which exists of a total of 2 301 instances spread over 12 classifiers and is split between the testing data and training data of the classifiers. We encounter a total of 61 critical terminations (either a time-out, memory limit, or other runtime error) over the testing and training queries combined, which is 2.6 % of the cases. In all other cases, we were able to estimate a lower-bound on the critical epsilon that is as tight as possible, given the discretisation of the problem.

Discretised epsilon values: We choose a range of epsilons based on what we have encountered in literature. Based on the results we conclude that this range seems reasonable as in none of the verification queries we have found a critical epsilon of 0.4. The maximum critical epsilon was 0.203 and the minimum



(a) (b)

Fig. 4: Cumulative distribution of the fraction of instances for which the $\tilde{\varepsilon}^*$ has been found for the training and testing data, for two different networks.

was 0.001. There were no cases where verifying an instance with an epsilon of 0.001 returned an adversarial example.

Connection with theory: The theoretical maximum number of verification subqueries is 11, as mentioned in Section 4.1. This theoretical maximum applies when there is a maximum of one missing answer, where we have an unknown number of missing answers. Empirical analysis shows that there was a maximum number of 18 verification sub-queries and an average number of 11.69 overall experiments performed. This average implies that the assumption of having at most one missing answer makes the problem easier, and that therefore, there is room for more theoretical work that studies the more complicated case where there can be an arbitrary number of missing answers.

5 Discussion and outlook

In this preliminary study, we have shown the limitations of looking at robust accuracy with respect to a fixed perturbation radius in the context of defining the robustness of a neural network. In particular, the relative robustness of networks varies greatly under different perturbation radii. As an alternative, we propose the critical epsilon value, which provides a far more nuanced view of neural network robustness. Rather than a single adversarial accuracy value, we now obtain a more informative distribution describing the robustness across observations. While determining critical epsilon values comes at an additional computational cost, we show that when utilising parallel k-binary search, they can be determined efficiently.

In particular, analysing these critical robustness distributions, we have found evidence that they closely resemble log-normal distributions. We also found

evidence that the robustness distribution for inputs that the network was not trained on (testing data) is not significantly different than that for the inputs used for training. Using these distributions, it becomes possible to confidently make general statements of the robustness of a given network, without the need to determine the critical epsilon value for the entire dataset. Further investigation is required to validate this by testing this hypothesis on a larger and more diverse set of neural network verification problems.

In future work, we plan to explore ways of allowing verification algorithms to find critical epsilon values directly, rather than via a wrapper method such as k-binary search. Additionally, we will investigate how robustness distributions change when neural networks are trained with various adversarial training methods.

Acknowledgements

This research was partially supported by TAILOR, a project funded by EU Horizon 2020 research and innovation program under GA No. 952215. The authors would like to thank Hadar Shavit and Bram Renting for useful discussions.

References

1. Bastani, O., Ioannou, Y., Lampropoulos, L., Vytiniotis, D., Nori, A., Criminisi, A.: Measuring Neural Net Robustness with Constraints. In: Advances in Neural Information Processing Systems 29 (NeurIPS 2016). pp. 2613–2621 (2016)
2. Botoeva, E., Kouvaros, P., Kronqvist, J., Lomuscio, A., Misener, R.: Efficient Verification of ReLU-based Neural Networks via Dependency Analysis. In: Proceedings of the 34th AAAI Conference on Artificial Intelligence (AAAI-20). pp. 3291–3299 (2020)
3. Bunel, R., Turkaslan, I., Torr, P., Kohli, P., Mudigonda, P.K.: A Unified View of Piecewise Linear Neural Network Verification. In: Advances in Neural Information Processing Systems 31 (NeurIPS 2018). pp. 1–10 (2018)
4. Cicalese, F., Gargano, L., Vaccaro, U.: On Searching Strategies, Parallel Questions, and Delayed Answers. Discrete applied mathematics **144**(3), 247–262 (2004)
5. Cicalese, F., Vaccaro, U.: Binary Search with Delayed and Missing Answers. Information processing letters **85**(5), 239–247 (2003)
6. De Palma, A., Bunel, R., Desmaison, A., Dvijotham, K., Kohli, P., Torr, P.H., Kumar, M.P.: Improved branch and bound for neural network verification via lagrangian decomposition. In: arXiv preprint arXiv:2104.06718 (2021)
7. Dvijotham, K., Stanforth, R., Gowal, S., Mann, T.A., Kohli, P.: A Dual Approach to Scalable Verification of Deep Networks. In: Proceedings of the 38th Conference on Uncertainty in Artificial Intelligence (UAI 2018). pp. 550–559 (2018)
8. Ehlers, R.: Formal Verification of Piece-Wise Linear Feed-Forward Neural Networks. In: Proceedings of the 15th International Symposium on Automated Technology for Verification and Analysis (ATVA 2017). pp. 269–286 (2017)
9. Gehr, T., Mirman, M., Drachsler-Cohen, D., Tsankov, P., Chaudhuri, S., Vechev, M.: AI2: Safety and Robustness Certification of Neural Networks with Abstract Interpretation. In: Proceedings of the 39th IEEE Symposium on Security and Privacy (IEEE S&P 2018). pp. 3–18 (2018)

10. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. arXiv preprint arXiv:1412.6572 (2014)
11. Henriksen, P., Lomuscio, A.: Efficient Neural Network Verification via Adaptive Refinement and Adversarial Search. In: Proceedings of the 24th European Conference on Artificial Intelligence (ECAI 2020). pp. 2513–2520 (2020)
12. Katz, G., Barrett, C., Dill, D.L., Julian, K., Kochenderfer, M.J.: Reluplex: An Efficient SMT Solver for Verifying Deep Neural Networks. In: Proceedings of the 29th International Conference on Computer Aided Verification (CAV 2017). pp. 97–117 (2017)
13. König, M., Bosman, A.W., Hoos, H.H., van Rijn, J.N.: Critically Assessing the State of the Art in CPU-based Local Robustness Verification. In: Proceedings of the Workshop on Artificial Intelligence Safety 2023 (SafeAI 2023) co-located with the Thirty-Seventh AAAI Conference on Artificial Intelligence (AAAI2023) (2023)
14. Li, L., Xie, T., Li, B.: Sok: Certified robustness for deep neural networks. In: 2023 IEEE Symposium on Security and Privacy (SP 2023). pp. 94–115. IEEE Computer Society (2023)
15. Liu, J., Chen, L., Miné, A., Wang, J.: Input validation for neural networks via runtime local robustness verification. In: arXiv preprint arXiv:2002.03339 (2020)
16. Müller, M.N., Brix, C., Bak, S., Liu, C., Johnson, T.T.: The third international verification of neural networks competition (vnn-comp 2022): Summary and results. In: arXiv preprint arXiv:2212.10376 (2022)
17. Singh, G., Gehr, T., Mirman, M., Püschel, M., Vechev, M.: Fast and effective robustness certification. In: Advances in Neural Information Processing Systems (NeurIPS 2028. vol. 31 (2018)
18. Singh, G., Gehr, T., Püschel, M., Vechev, M.: Boosting robustness certification of neural networks. In: Proceedings of the 7th International Conference on Learning Representations (ICLR 2019) (2019)
19. Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I.J., Fergus, R.: Intriguing properties of neural networks. In: Proceedings of the 2nd International Conference on Learning Representations (ICLR 2014). pp. 1–10 (2014)
20. Tjeng, V., Xiao, K., Tedrake, R.: Evaluating Robustness of Neural Networks with Mixed Integer Programming. In: Proceedings of the 7th International Conference on Learning Representations (ICLR 2019). pp. 1–21 (2019)
21. Wang, S., Pei, K., Whitehouse, J., Yang, J., Jana, S.: Efficient Formal Safety Analysis of Neural Networks. In: Advances in Neural Information Processing Systems 31 (NeurIPS 2018). pp. 6369–6379 (2018)
22. 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 neural network robustness verification. In: Advances in Neural Information Processing Systems (NeurIPS 2021). vol. 34, pp. 29909–29921 (2021)
23. 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: Proceedings of the 7nd International Conference on Learning Representations ICLR 2018 (2018)
24. Xiang, W., Tran, H.D., Johnson, T.T.: Output Reachable Set Estimation and Verification for Multilayer Neural Networks. In: IEEE Transactions on Neural Networks and Learning Systems. vol. 29, pp. 5777–5783 (2018)
25. Yang, Y.Y., Rashtchian, C., Zhang, H., Salakhutdinov, R.R., Chaudhuri, K.: A closer look at accuracy vs. robustness. In: Advances in Neural Information Processing Systems (NeurIPS 2020. vol. 33, pp. 8588–8601 (2020)