Annales Mathematicae et Informaticae
URL: https://ami.uni-eszterhazy.hu
Manuscript, February 2, 2024

Interval Based Verification of Adversarial Example Free Zones for Neural Networks – Dependency Problem

Tibor Csendes[a]

[a]University of Szeged, Institute of Informatics csendes@inf.szte.hu

Abstract. Recent machine learning models are sensitive to adversarial input perturbation. That is, an attacker may easily mislead an otherwise well-performing image classification system by altering some pixels. It is quite challenging to prove that a network will have correct output when changing slightly some regions of the images. This is why only a few works targeted this problem. Although there are an increasing number of studies on this field, really reliable robustness evaluation is still an open issue. We will present some theoretical results on the dependency problem of interval arithmetic what is critical in interval based verification.

Keywords: Verification Artificial neural network Interval arithmetic

AMS Subject Classification: AMS Subject Classifications 65G40, 68T07

1. Introduction

Szegedy et al. [7] showed first the phenomenon of adversarial examples. Since then the efforts for verification algorithms were concentrated around optimized models of the trained neural networks [9]. We were able to prove that these verification algorithms are unfortunately not reliable [11]. In the last ICAI conference we reported our first results with an interval based verification algorithm [4]. This approach applied simple natural interval extension for the calculation of inclusion functions, and we were able to produce realistic size adversarial example free zones for simple networks, that had good accuracy for the MNIST picture database distinguishing the hand written figures for the digits 3 and 7.

In the present work we report on our new results. The algorithm was reimplemented in the Julia language [8]. Our computational test results on the full

Table 1. The number of critical cases, for which at least one of the ReLU activation functions had an input interval containing zero – as a function of the interval width.

	interval width							
	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}	10^{-6}	10^{-7}	10^{-8}
No. critical cases	9178	2389	310	32	4	1	1	0

MNIST database of 10 different hand written digits with a modest but realistic size network (1 hidden layer of 128 neurons) could reach more modest verification results as those reported in [4]. We are also dealing with full-scale machine learning models: an artificial neural network based on the huBERT language model, trained to recognize Hungarian fake news in health-related texts, which contains approximately 110 million parameters. This is obviously out of reach for verification. On the other hand long standing open mathematical problems were solved by interval based computer aided methods, let us just mention our contribution to the solution of the Wright conjecture [2], and proving that the forced damped pendulum is chaotic [1].

The limitations of the naive interval arithmetic approach were felt already in our full MNIST test. It turned out that the critical question is how many activation functions obtain such an input interval that contains zero. If the inputs are real numbers then the probability of having a zero input for a ReLU activation function is in general zero for well trained networks. Table 1 gives the number of test cases from the picture database for which at least one of the ReLU activation functions had an input interval that contained zero. The moral of the results obtained is that the simple application of the algorithm introduced in [3] the application of which for neural network verification was reported in [4] is not sufficient, more sophisticated algorithms should be developed. As a preparatory step in that direction in the present paper a qualitative description is given on the possible amount of overestimation caused by the dependency problem of naive interval arithmetic.

2. Dependency Problem for Interval Calculations

We consider the fully connected simple feedforward artificial neural network having an input layer, one or more hidden layers and an evaluation output layer. We use the ReLU activation function $\max(0, x)$. This is a fairly general framework, and our results can be transferred in a more or less straightforward way to other types of neural networks such as convolutional and recurrent ones. In our model each neuron number k is defined by the function

$$y_k = \max\left(0, \sum_{i=0}^{n} w_i x_i\right)$$

of its inputs x_i with weights w_i, and $x_0 = 1$. Here x_i stands for one of the n outputs of the earlier layer, or that of the inputs. Each neuron has as inputs all outputs of the previous layer, the first layers input is obviously the input of the network.

Our neural network can be seen as a multidimensional function of the inputs for each output. Without the ReLU activation functions these functions would be simple linear combinations of the input values, in which the product of respective weights give the coefficients of the linear combinations. The ReLU activation functions cause some parts of the computation tree disappear due to their zero branch. The whole neural network forms the surface of a polygon in the high dimensional space of input variables.

Verification of artificial neural networks means that for a given input we can prove that for a neighbourhood of that input, we obtain the same classification for each point inside the tested neighbourhood. In other words, the outputs of the neural network should not change that much which would result in a different class. Hence the range of the function should not change much inside the neighbourhood of the given input. Interval calculations provide a straightforward tool for bounding the range of functions reliably.

2.1. Interval calculation

Consider $\mathbb{I}$ the set of compact real intervals containing intervals $x = [a, b]$, where $a, b, \in \mathbb{R}$, $a \leq b$. The four basic operations can easily be calculated for intervals based on the real calculations on the real end points of the argument intervals:

$$\begin{aligned}
[a, b] + [c, d] &= [a + c, b + d], \\
[a, b] - [c, d] &= [a - d, b - c], \\
[a, b] \cdot [c, d] &= [\min(ac, ad, bc, bd), \max(ac, ad, bc, bd)], \\
[a, b]/[c, d] &= [a, b] \cdot [1/d, 1/c] \text{ if } 0 \notin [c, d].
\end{aligned}$$

Not only the basic operations, but also standard functions like sin, *log* etc. can easily be generalized for interval arguments. It is important that we have the inclusion property: $f(x) \in F(X)$, where the real number x is in the interval X. This property holds also for careful computer implementations, which use outside rounding to have rigorous bounds on the range $f(X)$ of the real function $f(x)$. The above arithmetic rules ensure sharp bounds on the resulting reals. The numeric effect of the outside rounding is usually negligible. Still, interval arithmetic keeps the inclusion property.

The main difficulty in applying interval arithmetic in the evaluation of trained neural network lies in the so-called dependency problem. In spite of the fact that addition and multiplication gives sharp bound for intervals, the hidden dependencies of input variables pose a substantial problem in terms of overestimation of the bounded ranges [6]. To illustrate it consider the inclusion function $F(X)$ of the function

$$f(x) = x^2 - x.$$

It gives $F([0,1]) = [-1,1]$, while the range of it, $f(X)$ is here just $[-0.25, 0.0]$. The width of the calculated inclusion $w(F(X)$ is eight time as large as the width of the range $w(f(X))$. It is not the last word, using more sophisticated techniques the problem of the too loose enclosure can be overcome – at the cost of higher computing times.

For example affine arithmetic and the interval propagation technique [10] can help. In the present work a description is given on the possible effect of the dependency problem on the overestimation of interval evaluation of trained artificial neural networks.

2.2. Results

In the following, we investigate the overestimation amounts we can face while evaluating trained fully connected feed forward networks with the ReLU activation function. We study the situation when the weights of such a network are given as real numbers, we fix an input (e.g. a picture), and we test how large intervals around the input values can be verified to result in the same classification we obtained for the real case. In the next theoretical investigations we assume that interval arithmetic is calculated in the precise way, i.e. we exclude the effect of outside rounding. Note that only the dependency problem of the addition and subtraction should be considered for artificial networks, because the multiplication or division of input values do not play a role in the calculation of the output values.

Assertion 2.1. *For a fully connected feed forward standard artificial neural network the overestimation size $w(F(X)) - w(f(X))$ of the inclusion function can be zero only if at least one of the following conditions are fulfilled:*

- *all input intervals are of zero width: $w(x_i) = b - a = 0$,*
- *for all input variables x_i in the computation of each of the outputs all weights of them are of the same sign: either all nonnegative, or all nonpositive, and*
- *all the final evaluation functions calculating the outputs of the network have negative arguments.*

These conditions are not only sufficient one by one, but a proper combination of them is also necessary.

Proof. If the input intervals are of zero width, then the possible harmful effect of dependency has no room, since the lower and the upper bounds of all input intervals are equal. In this way the weighted subtraction results in the same value, indifferent of which bounds we choose from the arguments.

The second condition is sufficient, since in this case it is the same which output we calculate, and also for all input variables, the weights have the same sign, and in this way the dependency problem cannot happen for their weighted summation.

If the third condition is met, then all ReLUs providing the output variables must have the value of interval zero ($[0,0]$). It is exactly the range of the network for the input studied.

The obvious effect of a ReLU activation function is, that all possible combinations of the subnetwork that produce its negative argument for the output disappear. In other words, we obtain an equivalent network for the given input, if we neglect the subnetwork producing the negative argument of the given ReLU activation function. If we evaluate our network on input intervals in stead of input real numbers, then the whole argument interval of the given ReLU activation function must be negative to have this feature. The remaining evaluation network will be sensitive for the first two earlier defined conditions, and if none holds, then the dependency problem will necessarily corrupt the result. Other combinations of the three conditions are possible that together ensure that the given neural network on the considered input will not produce positive overestimation size. □

The conditions in Assertion 2.1 are quite strong, and in full-size neural networks capable to solve real life problems, these can hardly be met. Note that the case when only a single hidden layer is in the network is implicitly covered by the second condition, since then each input interval will only be multiplied by a single weight. Single hidden layer fully connected feed forward neural networks seem to be too simple, but they have full recognition capacity with proper activation to get the output – at the cost of a large number of neurons in that single layer [5].

Consider now the question which are the major factors for the overestimation sizes in the same setting.

Assertion 2.2. *For a fully connected feed forward standard artificial neural network of k input intervals, m neurons in each of the even number of n hidden layers, and all weights w_i are bounded by $|w_i| \leq W$ the amount of overestimation $w(F(X)) - w(f(X))$ of the inclusion function of an output is not more than $2^{n/2} m^{n/2} W^n \sum_{i=1}^{k} w(X_i)$.*

Proof. As a consequence of Assertion 2.1 to have positive overestimation we must have positive width input intervals, two weights for their addition with opposing signs, and a path to output values through ReLU activation functions that have arguments which have positive values as well. Since our feed forward network is fully connected, in each neuron we have every interval from the previous layer multiplied by weights. No overestimation can occur in the first hidden layer, since here each input is just multiplied by a weight, but no subtraction of the same variable may be active.

One neuron on the second hidden layer can produce at most $2W^2 w(X_i)$ overestimation, where $w(X_i)$ is the width of the original input interval X_i, and W is the upper bound of the absolute value of the weights in the network. This is the case, when $w_l w_j X_i - w_l w_j X_i$ is in the actual sum of the neuron. The range of this sum is zero, and in this way the overestimation size is $2W^2 w(X_i)$.

Consider now the case when all weights of the network that will affect this overestimation are of the same sign. Then the calculated overestimation changes also by subsequent multiplications by weights as we calculate the output values. The isotonicity property of interval arithmetic implies that this overestimation cannot disappear in the evaluation of the network — with the exception of multiplication

by zero. For this kind of overestimation we obtain the upper bound of $2W^n w(X_i)$. For each input variable and neuron we can calculate with similar overestimation, that sums up to $2mW^n \sum_{i=1}^{k} w(X_i)$. According to this expression, the type of networks we discussed in the previous paragraph produce two times larger overestimation compared to fix sign weights considered in the present paragraph.

In all other scenarios the overestimation generated in the two first layers will be strengthened by the similar effects of the dependency problem, but this time the outputs of the earlier level neurons will play the role of the input intervals. The respective overestimation obtained on an output of a subsequent two level part of the network is $2mW^2 \sum_{i=1}^{m} w(Y_i)$, where Y_i are the output intervals of the preceding part of the network.

The total overestimation amount can be calculated for an output value of a deep neural network as

$$2^{n/2} m^{n/2} W^n \sum_{i=1}^{k} w(X_i),$$

for even integers n. For odd n-s this upper bound is accordingly smaller by the amount of the last overestimation caused by the effect of the dependency problem on the top layer. □

Example 2.1. To illustrate the overestimation caused by the dependency problem in interval based verification of artificial neural networks, consider a simple network having two neurons in the first layer and the ReLU activation functions after them, and one more layer having a single neuron. All neurons have zero bias for simplicity.

The function that describes it effect is

$$f(x) = w_3 \max(0, w_1 x) + w_4 \max(0, w_2 x)).$$

Fix the weights to $w_1 = w_2 = 1$ and $w_3 = 1$, $w_4 = -1$. Let's calculate first the inclusion function of $f(x)$ for the argument interval $[0, 1]$:

$$F([0,1]) = ReLU([0,1]) - ReLU([0,1]) =$$
$$ReLU([0,1]) - ReLU([0,1]) = [0,1] - [0,1] = [-1,1].$$

Here, for the sake of simplicity, the same notation was used for the interval version of the ReLU function as for the usual, real one. Otherwise, the interval version of ReLU can be e.g. $[\max(0, a), \max(0, b)]$ for an argument interval of $[a, b]$. The correct range of this network as a real function is $[0,\ 0]$. The amount of overestimation is now $2 - 0 = 2$. This is in accordance with Assertion 2.2, and shows that the upper bound given there is sharp (with $m = 2$, $n = 2$, $W = 1$, and $k = 1$).

Corollary 2.2. *A direct consequence of Assertion 2.2 is that we can have the same amount of overestimation due to the dependency problem with decreasing the number of hidden layers while increasing the number of neurons in a layer and vice versa.*

3. Conclusion

Interval arithmetic based methods are promising for the verification of artificial neural networks, but the dependency problem is a serious threat, and it should be handled carefully to obtain reasonable size adversarial example free zones in acceptable amount of computation time. We have characterized those few cases when the dependency problem does not corrupt the inclusion function of a network, and described how the parameters (input size, weight parameter bound, number of layers and neurons in the layers) affect the overestimation of interval inclusion functions.

The main consequence of our theoretical study is that we can control the amount of overestimation caused by the dependency effect of interval arithmetic by forcing advantageous parameters such as low absolute bound of weights, or minimizing the number of hidden layers – while keeping the expected level of precision and recall. Still, the obvious proven full solution for the dependency problem, a single hidden layer with proper activation to get the output, is probably computationally too complex to be applicable.

Acknowledgements. This research was supported by the project Extending the activities of the HU-MATHS-IN Hungarian Industrial and Innovation Mathematical Service Network EFOP3.6.2-16-2017-00015, 2018-1.3.1-VKE-2018-00033, and by the Subprogramme for Linguistic Identification of Fake News and Pseudoscientific Views, part of the Science for the Hungarian Language National Programme of the Hungarian Academy of Sciences (MTA). The author acknowledges the implementation and testing work done by Soma Timer.

References

[1] B. Bánhelyi, T. Csendes, B. Garay, L. Hatvani: *A computer-assisted proof for Sigma_3-chaos in the forced damped pendulum equation*, SIAM J. on Applied Dynamical Systems 7 (2008), pp. 843–867, doi: https://doi.org/10.1137/070695599.

[2] B. Bánhelyi, T. Csendes, T. Krisztin, A. Neumaier: *Global attractivity of the zero solution for Wright's equation*, SIAM J. on Applied Dynamical Systems 13 (2014), pp. 537–563, doi: https://doi.org/10.1137/120904226.

[3] T. Csendes: *An interval method for bounding level sets of parameter estimation problems*, Computing 41 (1989), pp. 75–86, doi: https://doi.org/10.1007/BF02238730.

[4] T. Csendes, N. Balogh, B. Bánhelyi, D. Zombori, R. Tóth, I. Megyeri: *Adversarial Example Free Zones for Specific Inputs and Neural Networks*, in: Proceedings of the 2020 ICAI, Eger, Hungary, 2020, url: https://ceur-ws.org/Vol-2650/paper9.pdf.

[5] G. Cybenko: *Approximation by superpositions of a sigmoidal function*, Mathematics of Control, Signals, and Systems 2 (1989), pp. 303–314, doi: https://doi:10.1007/BF02551274.

[6] H. Ratschek, J. Rokne: *Computer Methods for the Range of Functions*, Chichester: Ellis Horwood, 1984, doi: https://doi.org/10.2307/2008155.

[7] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, R. Fergus: *Intriguing properties of neural networks*, in: Proceedings of the 2014 International Conference on Learning Representations, 2014, doi: https://doi.org/10.48550/arXiv.1312.6199.

[8] S. Timer: *Interval method for the verification of artificial neural networks in Julia language (in Hungarian)*, Szeged: University of Szeged, BSc Dissertation, 2023.

[9] V. Tjeng, K. Xiao, R. Tedrake: *Evaluating Robustness of Neural Networks with Mixed Integer Programming*, in: Proceedings of the 2019 International Conference on Learning Representations, 2019, doi: https://doi.org/10.48550/arXiv.1711.07356.

[10] S. Wang, K. Pei, J. Whitehouse, J. Yang, S. Jana: *Formal security analysis of neural networks using symbolic intervals*, in: Proceedings of the 27th USENIX Conference on Security Symposium, SEC'18, 2018, pp. 1599–1614, url: https://doi.org/10.48550/arXiv.1804.10829.

[11] D. Zombori, B. Bánhelyi, T. Csendes, I. Megyeri, M. Jelasity: *Fooling a complete neural network verifier*, in: Proceedings of the 2021 International Conference on Learning Representations, 2021, url: https://openreview.net/forum?id=4IwieFS44l.