Proceedings of the 11th International Conference on Applied Informatics
Eger, Hungary, January 29–31, 2020, published at http://ceur-ws.org

Adversarial Example Free Zones for Specific Inputs and Neural Networks

Tibor Csendes[a], Nándor Balogh[b], Balázs Bánhelyi[a], Dániel Zombori[a], Richárd Tóth[a], István Megyeri[a]

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

[b]Redink Ltd., Szeged, Hungary
bn@redink.hu

Abstract

Recent machine learning models are highly sensitive to adversarial input perturbation. That is, an attacker may easily mislead a well-performing image classification system by altering some pixels. However, proving that a network will have correct output when changing some regions of the images, is quite challenging. Because of this, only a few works targeted this problem. Although there are an increasing number of studies on this field, reliable robustness evaluation is still an open issue. In this work, we will attempt to contribute in this direction. We will present new interval arithmetic based algorithms to provide adversarial example free image patches for trained artificial neural networks.

Keywords: artificial neural networks, adversarial example, interval arithmetic, inscribed interval

MSC: 68T05, 65G30

1. Introduction

One of the hottest topics in present artificial intelligence research is to understand the phenomenon of adversarial examples for machine learning techniques applying artificial neural networks [7, 15]. The typical such image classification problem is the following. After the proper training of the network, there exist pictures surprisingly similar to the positive sample images that result in a wrong denial

decision. As an illustration of the problem, see some real life images of car licence plates on Figure 1, that could not be detected correctly.

Figure 1: Real life examples with car licence plate number classification problems

Using adversarial examples generated by existing attack algorithms like those in [11, 4, 13], an offending driver can easily prevent the system to identify him. Even in black box cases, i.e. when the attacker has no access to model parameters, the attack can be successful [12]. This makes it difficult to apply these state-of-the-art techniques in any safety critical settings. One might naively use attack algorithms for evaluating robustness. However, as show in [4], robustness against some attacks does not mean that the network is robust. Later, stronger attacks may be developed, which will be able to fool the network. A certified evaluation may end this arms race. Further, it motivates to develop reliable methods for evaluating neural networks.

In this paper we present our first results on implementing an interval arithmetic based reliable algorithm to describe adversarial example free zones on an image for trained artificial neural networks.

2. Verified computation

There are already many available shocking results regarding adversarial examples for artificial neural networks (see e.g. those in [8, 13, 18]). Also, many approximate procedures are suggested for e.g. locating the nearest adversarial example to a given correctly accepted image. On the other hand, we do not know about existing verified implemented techniques being capable of providing adversarial example free zones. Obviously there are approaches in this direction [6, 10, 16, 17]. This latter feature is important for mathematically correct statements, especially on a field, where the expected behavior of a computational method differs sometimes from the anticipated one. Interval arithmetic based verified numerical calculations are the proper tool for handling both rounding errors and their consequences, and also for proving statements on positive measure sets of high dimension. We applied interval methods to prove that the damped forced pendulum is chaotic [2], we proved most of the Wright conjecture on a delayed differential equation [3], and verified new optimal circle packing instances [14].

The set theoretical definition of interval arithmetic is:

$$A \circ B = \{a \circ b \mid a \in A \text{ and } b \in B\},\ A, B \in \mathbb{I},$$

where $\mathbb{I}$ is the set of compact intervals $[i, j]$, where $i, j \in \mathbb{R}$ and $i \leq j$.

The arithmetic definition is:

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

These definitions are equivalent. Although the obtained result intervals seem to be sharp, it is not the case. The inclusion of the function

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

obtained for the interval $[0,1]$ is $[-1,1]$, while the range of the function $f(x)$ (the set of possible values) is here just $[-0.25,0.0]$. Using more sophisticated techniques the problem of the too loose enclosure can be overcome – at the cost of higher computing times.

In a floating point environment (most cases) the *outward rounding* is important to have a conservative inclusion that is a must in computer supported mathematical proofs. Outward rounding means that the bounds of the calculated result intervals are rounded in such a way that all result points are within the given bounds. In other words, the lower bound is always rounded toward $-\infty$, and the upper bound toward ∞. This can easily be realized applying the four rounding modes of the IEEE 754 standard (available on most programming languages and computers). Several programming languages and packages support interval arithmetic based inclusion function generation: C-XSC, FORTRAN-XSC, PASCAL-XSC, and PROFIL/BIAS. Interval packages are available in several symbolic and numerical systems such as Maple, Mathematica, Matlab. The package for the latter one, Intlab is especially easy to use.

It is important to note that interval calculations do not require the knowledge of the symbolic expression of the underlying functions, it is enough if a computer subroutine is available. In our case a Python code was applied. Linear sums and sigmoid or other monotonous functions used in artificial neural networks are explicitly advantageous for interval inclusion functions: sharp bounding is expected in general. On the other hand, non-monotonic activations can also be handled by the presented method.

For our problem, we need a proper method to describe the large dimensional sets that cannot contain adversarial examples. For this purpose, an interval arithmetic based algorithm describing the level sets of nonlinear optimization problems [5] seems to be appropriate. Unfortunately, this algorithm scales up very badly with increasing dimension. This is why first we developed interval arithmetic based algorithms that are capable of describing the level sets of an artificial neural network around a feasible positive sample. In this way, we could ensure with mathematical rigor that adversarial samples cannot exist within the found bounds. According to our experiences, benevolent problems show much better complexity numbers compared to theoretically possible pessimistic convergence rates.

3. Results

The simple, logistic regression model were trained on the subset of the MNIST dataset [9] that includes two classes: 3 and 7. We used 10 images from this database which contain 28×28 pixel grayscale images of handwritten digits. Assuming n examples (x_i, y_i), $x_i \in \mathbb{R}^n$, $y_i \in \{0, 1\}$, $i = 1, \ldots, n$, the goal is to approximate the data using the logistic function $y \approx \sigma(w^T x + b) = 1/(1 + e^{-w^T x + b})$.

We used the negative log likelihood loss function

$$L(w, b) = -\sum_{i=1}^{n} y_i \cdot \log(\sigma(x_i; w, b)) + (1 - y_i) \cdot \log(1 - \sigma(x_i; w, b))$$

to find the best model (that is, w and b). As our optimizer, we used ADAM [1] with a minibatch size of 32 for 100 epochs.

The Python code of the obtained network was used for the evaluation. We have applied the Python interval package on a simple 1.8 GHz Intel core i5 processor laptop under Windows 7, within the PyCharm development system.

3.1. Changes on the whole picture

First we checked how much we can change the actual grayscale values of a picture without having an adversarial example case. It means, that for each pixel we allowed a given amount of relative change in the grayscale values. E.g. 1% means an alteration of 3 for a pixel that has the white of the value 255. We understand the relative change in the neighborhood of zero still in an absolute way, i.e. if the given pixel was black with the grayscale value of zero, then in our calculation the interval [0, 1] was actually checked. Note that this way of adding noise to a picture is realistic in the sense, that many practical situations can fall into this category, including for example traffic sign pictures in a slightly foggy weather. Also, many documented adversarial examples were obtained by added random noise, where the relative change in the grayscale values were limited.

We composed a simple greedy algorithm to find the largest possible relative difference value efficiently. Here n is the number of pixels; p is the picture pixels in a vector; *maxpercent* is the proven number of percents, initially set to zero; *percent* is the number of percent of changes to be checked, first it is set to one; $p0$ is the starting picture; *greater* is a boolean variable meaning whether the network gives a value above 0.5: then it is true, otherwise false. The simplified pseudo code of the algorithm is:

0. If $F(p0) > 0.5$ then $greater = true$, otherwise $greater = false$
1. Iterate until $percent <= 100$
2. Let P be an n dimensional interval
3. For $i = 1$ to n do

(a) If $p_i = 0$, then $P_i = [0, 2 * percent/100]$

(b) Otherwise, if $p_i = 1$, then $P_i = [1 - 2 * percent/100, 1]$

(c) Otherwise $P_i = [p_i - percent/100, p_i + percent/100]$, and check the end points: if the lower one is negative, then set it to zero, if the upper one is larger than 1, then set it to 1.

4. If $greater = true$ and $F(P) \geq 0.5$, or $greater = false$ and $F(P) < 0.5$ then do:

   (a) If $percent < 1$, then $maxpercent = percent$, and break the main cycle, Stop.

   (b) Otherwise $maxpercent = percent$, and $percent = percent + 1$

5. Otherwise if $percent < 1$, then set $percent = percent - 0.1$

   (a) If now $percent = 0$, then set $maxpercent = 0$ and STOP

   (b) Otherwise break the outer loop

6. End of the cycle started in the first step

From computational point of view, the above described checking means a single interval evaluation of a trained network, when instead of the usual real number grayscale values, real compact intervals should be evaluated. Since interval calculation is according to the rule of thumb 4-35 times slower than the respective real number calculations, this type of adversary example free set checking does not require long computation. This is why we have composed a simple algorithm that will increase the size of the checked interval until the respective conditions are hurt. Our measured computation times for 10 images was 5.05 seconds. Note that if two possible values form the interval box in each dimension, then one interval evaluation will prove that all the 2^{784} points satisfy the condition set by the trained artificial neural network. This is obviously not to be completed with one-by-one real evaluations.

The proven amount of changes on the gray scale values *everywhere* on Figure 2 without having an adversarial example were in the order of appearance: 1%, 1%, 1%, and 2%, respectively. These proven values are useful in real life situations.

3.2. Arbitrary large changes in neighboring points

As a second try, we aimed to find those maximal rectangles in an image for which all pixels may change their grayscale value arbitrarily between 0 and 255 without being classified incorrectly. Basically, we grow squares around given pixels, check their recognizability, and if it was positive, then enlarge them. Again, we built a simple, efficient greedy algorithm for this purpose. Also, we take care of the sides of the original image, that is why our result may be a rectangle. We repeat our growing procedure for all the pixels of our image, and return that rectangle that had the most pixels.

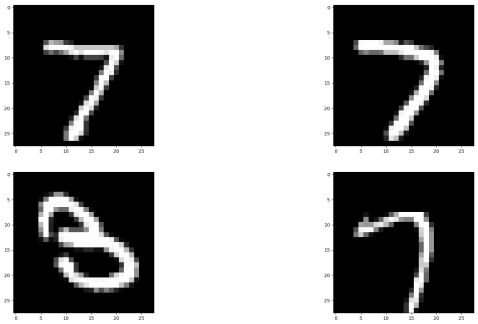

Figure 2: Handwritten number classification test problem instances from the MNIST test set

Denote the picture by p a vectorial form; n is the number of pixels; $maxpixels$ is the maximal number of pixels found in a rectangle; $p0$ is the starting picture; $greater$ is again a boolean variable meaning whether the network gives a value above 0.5: then it is true, otherwise false; $allpixel$ is a boolean variable meaning whether the pixels of the actual picture p can attain all possible values, the default value is true; and s is the size of the picture, for a 28×28 picture it is 28. The simplified pseudo code of the second algorithm is:

0. If $F(p0) > 0.5$ then $greater = true$, otherwise $greater = false$
1. For $i = 1$ to n do
2. Set $p_i = [0, 1]$.
   (a) If $greater = true$ and $\min F(p) \leq 0.5$, or $greater = false$ and $\max F(p) \geq 0.5$ then $allpixel = false$ and go to the next iteration round
   (b) Otherwise if $maxpixels = 0$ then set $maxpixels = 1$
3. Set $distance = 1$
4. Iterate the next 3 steps
5. Set $indices = pixel\_indices(i, distance, s, s)$
6. Set $p = modifypixels(p, indices)$
   (a) If $greater = true$ and $\min F(p) < 0.5$, or $greater = false$ and $\max F(p) \geq 0.5$, then break the iteration
   (b) Otherwise: if the length of the list $indices$ is larger than $maxpixels$, then set $maxpixels$ equal to the length of the list $indices$
7. Set $distance = distance + 1$

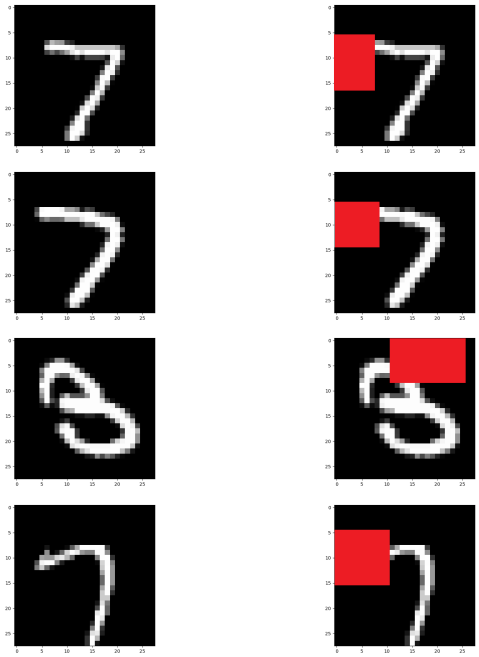

Figure 3: Original pictures and proven rectangles where we can change *everything* without having an adversarial example.

8. Set $p = pstart$, and continue with the next iteration

The obtained results are illustrated on Figure 3 for some of the studied images. The calculated number of pixels to be changed arbitrarily were between 88 and 190 (compare it with the $28 \times 28 = 784$ pixels in the images). The combined running time for the second round of 10 test images was 1971.87 second, i.e. closely half an hour.

4. Conclusion

We are still in the phase when we explore the capabilities of interval arithmetic based algorithms for describing the sensitivity of trained natural neural networks to changes in object to be classified, but we find our present results encouraging enough to continue our research project. The next issue can be the frightening case of small patches changing the recognized meaning of traffic signs. On the other hand, we could also be capable of proving where the next adversarial example is relative to a given image.

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. The authors are grateful for the anonymous referees for their useful suggestions to improve the paper.

References

[1] Ba, J. and Kingma, D., Adam: A Method for Stochastic Optimization. 3rd Intl. Conf. on Learning Representations (ICLR), 2015, http://arxiv.org/abs/1412.6980

[2] Bánhelyi, B., Csendes, T., Garay, B.M., and Hatvani, L., A computer-assisted proof for Sigma_3-chaos in the forced damped pendulum equation. *SIAM J. on Applied Dynamical Systems* Vol. 7. (2008), 843–867.

[3] Bánhelyi, B., Csendes, T., Krisztin, T., and Neumaier, A., Global attractivity of the zero solution for Wright's equation. *SIAM J. on Applied Dynamical Systems* Vol. 13 (2014), 537–563.

[4] Carlini, N. and Wagner, D.A.,, Towards Evaluating the Robustness of Neural Networks. *IEEE Symposium on Security and Privacy*, SP 2017, San Jose

[5] Csendes, T., An interval method for bounding level sets of parameter estimation problems, *Computing* Vol. 41 (1989), 75–86.

[6] Fazlyab, M., Morari, M., and Pappas, G.J., Safety Verification and Robustness Analysis of Neural Networks via Quadratic Constraints and Semidefinite Programming. *arXiv:*1903.01287v1.

[7] Goodfellow, I., Shlens, J., and Christian Szegedy, Explaining and Harnessing Adversarial Examples. *International Conference on Learning Representations*, 2015

[8] Ilyas, A., Santurkar, S., Tsipras, D., Engstrom, L., Tran, B., and Madry, A., Adversarial Examples Are Not Bugs, They Are Features. *arXiv:*1905.02175.

[9] Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P., Gradient-Based Learning Applied to Document Recognition, *Proc. of the IEEE*, 86 (1998) 2278–2324.

[10] Lin, W., Yang, Z., Chen, X., Zhao, Q., Li, X., Liu, Z., and He, J., Robustness Verification of Classification Deep Neural Networks via Linear Programming. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), IEEE Explore, 2019, 11418-11427, DOI: 10.1109/CVPR.2019.01168.

[11] Moosavi-Dezfooli, S.M., Fawzi, A., and Frossard, P.,, DeepFool: A Simple and Accurate Method to Fool Deep Neural Networks. *The IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016, 2574–2582.

[12] N. Narodytska and S. Kasiviswanathan, Simple Black-Box Adversarial Attacks on Deep Neural Networks. *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2017

[13] Su, J., Vargas, D.V., and Kouichi, S., One pixel attack for fooling deep neural networks. *arXiv:*1710.08864.

[14] Szabó, P.G., Markót, M.Cs., Csendes, T., Specht, E., Casado, L.G., and García, I., New Approaches to Circle Packing in a Square - With Program Codes, Springer, Berlin, 2007.

[15] Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I.J., and Fergus, R., Intriguing properties of neural networks. *International Conference on Learning Representations*, 2014

[16] Vincent Tjeng, V., Xiao, K., and Tedrake, R., Evaluating Robustness of Neural Networks with Mixed Integer Programming. *arXiv:*1711.07356v3.

[17] Xiang, W., and Johnson, T.T., Reachability Analysis and Safety Verification for Neural Network Control Systems. *arXiv:*1805.09944v1.

[18] Zaj, M., Zolna, K., Rostamzadeh, N., and Pinheiro, P.O., Adversarial Framing for Image and Video Classification. *The Thirty-Third AAAI Conference on Artificial Intelligence (AAAI-19).*