Verification of Sigmoidal Artificial Neural Networks using iSAT

Dominik Grundt, Sorin Liviu Jurj
OFFIS - Institute for Information Technology
Oldenburg, Germany
dominik.grundt@offis.de, sorin.jurj@offis.de

Willem Hagemann, Paul Kröger,
Martin Fränzle
Carl von Ossietzky University
Oldenburg, Germany
willem.hagemann@uol.de, paul.kroeger@uol.de,
martin.fraenzle@uol.de

This paper presents an approach for verifying the behavior of non-linear Artificial Neural Networks (ANNs) found in cyber-physical safety-critical systems. More exactly, we implement a dedicated sigmoid function propagator for the interval constraint propagation used in the satisfiability modulo theories solver iSAT. Experimental results show that the proposed solution for verifying non-linear ANNs using interval constraint propagation is more efficient regarding runtime and scalability on different models when compared to existing approaches.

This work received funding from the German Federal Ministry of Economic Affairs and Energy (BMWi) through the KI-Wissen project under grant agreement No. 19A20020M.

1 Introduction

In the age of highly automated systems and the development of autonomous systems, a possible application scenario for ANNs is to use them as controllers for cyber-physical safety-critical systems [5]. The goal of cyber-physical safety-critical systems is to capture the often complex environment, analyze the data and make control decisions about the future system behavior in such a way that it can guarantee safety, without endangering human life. Whenever such guarantees are obtained via formal verification of the system behavior, an ANN being a component of the system under analysis has also to be subject to verification [13]. However, although the internal composition of nodes and connections in ANNs seems to be suitable for classical verification methods, because of their non-linear activation functions, which are hard to analyze, these methods are usually not scalable and suited for verification [13].

One goal in the verification of cyber-physical safety-critical systems, including ANNs, is to verify their safety-critical properties. To this extent, satisfiability modulo theory (SMT) solvers are often used for proving that a requirement is satisfied for a feasible assignment of environmental variables [8]. Another important aspect is that, in order to combine both automatic verification and ANNs with non-linear activation functions, the solvers used must be able to handle non-linearity.

However, due to the complexity of ANNs, non-linear activation functions such as the sigmoid function can slow down the verification speed significantly [6]. In the literature, only restricted decidability results for the verification of ANNs with non-linear and transcendental activation functions can be found [5]. Such ANNs slow down the verification speed to such an extent that current solvers cannot handle it in a reasonable time, and are only able to verify ANNs consisting of around 20 nodes [6]. Regarding this, one of the promising approaches for verifying non-linear and transcendental activation function networks is the iSAT solver [12]. iSAT makes use of interval constraint propagation where the target is to contract intervals such that the constraint system is consistent throughout the box defined by the contracted intervals [4].

Submitted to:
7th International Workshop on
Symbolic-Numeric Methods for
Reasoning about CPS and IoT

The iSAT solver reasons over the — in general undecidable — theory of Boolean combinations of non-linear arithmetic constraints. It tightly couples the well-known Boolean decision procedure DPLL [3] with interval constraint propagation (ICP) [11] for real arithmetic. In addition to basic arithmetic operations, iSAT also supports non-linear and transcendental operators. Given an input formula and interval bounds of the real variables, iSAT searches for a satisfiable solution within the interval bounds.

The iSAT algorithm incorporates an alternation of deduction and decision steps. A decision step selects a variable, splits its current interval and decides for one of the resulting intervals to be the search space of the variable in the further course of the search. A deduction step then applies forward and backward interval constraint propagation until either a fixed point is reached, or an empty interval is derived which means that the formula is unsatisfiable under the current assumptions (decisions). The latter result causes a revoke of deductions of interval bounds and a reversal of decisions which yields, if possible, a decision for a yet unexplored part of the search space. If no decision for an unexplored part of the search space is possible, the formula is proven to be unsatisfiable under the initial assumption of the bounds on the search space. Since iSAT is able to deal with non-linear and transcendental functions, it seems to be a promising tool for verification of ANNs employing sigmoid functions as activation functions. ANNs such as Deep Learning (DL) architectures make use of the non-linear and transcendental sigmoid function (and their evolved variations) in hidden and output layers [9]. However, iSAT does currently not provide a dedicated interval constraint propagator for the sigmoid function or other non-linear activation functions.

In this paper we show an approach for verifying non-linear ANNs by making use of a dedicated sigmoidal interval constraint propagator integrated into an SMT solver. In our experiments, we demonstrate that the proposed approach improves not only runtime performance but also scalability, being a promising direction for the verification of non-linear ANNs.

2 Proposed Approach

As mentioned earlier, we aim at verification of ANNs using the sigmoid function as activation function. The sigmoid function $\mathrm{sig}(\cdot)$ is a bounded function from $\mathbb{R}$ to the open interval $(0,1) \subseteq \mathbb{R}$ and it is defined as follows:

$$\mathrm{sig}(x) := \frac{1}{1+\mathrm{e}^{-x}}.$$

The graph of the sigmoid function is shown in Fig. 1. As iSAT has no built-in sigmoid function, we

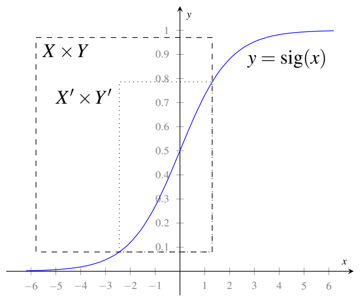

Figure 1: An example of an interval constraint propagation for the sigmoid function

used three different approaches to encode such ANNs into iSAT, namely a) an encoding of the sigmoid

Algorithm 1: $\mathsf{fwd\_prop}_\sigma(X,Y)$

```
Input: $X = (\bowtie_1, x_1, \bowtie_2, x_2)$, $Y$
$y_1 := \lfloor\sigma\rfloor(x_1)$;
$y_2 := \lceil\sigma\rceil(x_2)$;
$\bowtie_1' := \bowtie_1$;
$\bowtie_2' := \bowtie_2$;
if $y_1 = -\infty$ then $\bowtie_1' := <$;
if $y_2 = +\infty$ then $\bowtie_2' := <$;
$Y' := Y \cap (\bowtie_1', y_1, \bowtie_2', y_2)$;
return $Y'$;
```

Algorithm 2: $\mathsf{bwd\_prop}_\sigma(X,Y)$

```
Input: $X$, $Y = (\bowtie_1, y_1, \bowtie_2, y_2)$
$x_1 := \lfloor\sigma^{-1}\rfloor(y_1)$;
$x_2 := \lceil\sigma^{-1}\rceil(y_2)$;
$\bowtie_1' := \bowtie_1$;
$\bowtie_2' := \bowtie_2$;
if $x_1 = 0$ then $\bowtie_1' := <$;
if $x_2 = 1$ then $\bowtie_2' := <$;
$X' := X \cap (\bowtie_1', x_1, \bowtie_2', x_2)$;
return $X'$;
```

function in terms of a combination of operators available in iSAT (further referred as *composed approach*) s.t. an expression of the form $z = \mathrm{sig}(x)$ can equivalently be encoded as $1 = z(1 + \exp(-x))$, b) an approximative approach inspired by NeVer [10] where the sigmoid function is approximated into a piecewise-linear function for the interval [-8, 8] with a step size of 0.5, and c) a dedicated interval constraint propagator for the sigmoid function which we implemented into iSAT.

In the remainder of this section, we report on the implementation details of the sigmoid propagator. First, let us consider the defining property of a propagator of an unary function. Let f be an unary real function and $X \times Y$ the Cartesian product of two real intervals X and Y. We call $(x,y) \in X \times Y$ a *feasible solution for f in $X \times Y$* iff $y = f(x)$. The *interval constraint propagation* for f applied to $X \times Y$ yields the smallest subset $X' \times Y'$ of $X \times Y$ such that any feasible solution in $X \times Y$ is still a feasible solution in $X' \times Y'$. Fig. 1 depicts an example for an interval constraint propagation. To provide further implementation details, we note that an interval $I = \{x \mid x_1 \bowtie_1 x \bowtie_2 x_2\}$ can uniquely be written as the 4-tuple $(\bowtie_1, x_1, \bowtie_2, x_2)$, where $\bowtie_1, \bowtie_2 \in \{<, \leq\}$ denotes the boundary type and $x_1, x_2 \in \overline{\mathbb{R}}$ with $\overline{\mathbb{R}} := \mathbb{R} \cup \{-\infty, +\infty\}$ denotes the boundary values of the interval. We exploit that $\mathrm{sig}(\cdot)$ is continuous and strictly monotonic increasing. Hence, the interval bounds of X' and Y' of the propagation of $X \times Y$ for $\mathrm{sig}(\cdot)$ can in general be computed as the images and preimages of the boundary values of X and Y. The only special cases arise when the bounds of X are infinite, or when the bounds of Y are outside the domain of $\mathrm{sig}(\cdot)$. To this end, we extend the sigmoid function to $\overline{\mathbb{R}}$ and formally define $\sigma : \overline{\mathbb{R}} \to \overline{\mathbb{R}}$ and its inverse $\sigma^{-1} : \overline{\mathbb{R}} \to \overline{\mathbb{R}}$ as

$$\sigma(x) := \begin{cases} 0 & \text{if } x = -\infty \\ \frac{1}{1+\mathrm{e}^{-x}} & \text{if } -\infty < x < \infty \\ 1 & \text{if } x = +\infty, \end{cases} \qquad \sigma^{-1}(y) := \begin{cases} -\infty & \text{if } y \leq 0 \\ -\ln(\frac{1}{y} - 1)) & \text{if } 0 < y < 1 \\ +\infty & \text{if } 1 \leq y. \end{cases}$$

In iSAT, the propagation is split into two functions, the forward propagation step $\mathsf{fwd\_prop}_\sigma(X,Y)$ (Alg. 1) that returns the Y'-component, and the backward propagation step $\mathsf{bwd\_prop}_\sigma(X,Y)$ (Alg. 2) that returns the X'-component of the propagation $X' \times Y'$. As iSAT deals with floating point numbers, Alg. 1 and 2 refer to variants $\lceil\sigma\rceil$, $\lfloor\sigma\rfloor$, $\lceil\sigma^{-1}\rceil$, $\lfloor\sigma^{-1}\rfloor$ that implement a safe outward rounding of interval bounds towards the indicated direction.

3 Experiments & Results

We evaluate the newly implemented dedicated sigmoid propagator and iSAT with a focus on runtime and scalability. Based on the MNIST dataset [7] and a self-generated dataset of a simplified function of the

Table 1: Distribution of neurons in hidden layers of ETCS & MNIST networks (fully connected & feed-forward).

model name	no. of layers	neurons per layer	total no. of neurons
ETCS-small	4	3–9	27
ETCS-big	4	25	100
ETCS-1	1	500	500
ETCS-2	2	250	500
ETCS-4	4	125	500
ETCS-8	8	62–64	500
ETCS-16	16	31–35	500
ETCS-32	32	14–18	500
ETCS-64	64	6–10	500
ETCS-125	125	4	500
MNIST-small	2	(784, 392, 196, 10)	588 hidden, 1 382 total
MNIST-big	2	(784, 784, 392, 10)	1 176 hidden, 1 970 total

moving block authorization of the European Train Collision System (ETCS) [1], a total of 12 different neural network models were developed and trained using the Keras API [2]. In total, more than 150 experiments were generated, executed and analyzed with the neural network models.

The function modeled for the ETCS-based experiments provides an emergency brake advisory for a following scenario of two trains depending on the maximum permitted speed and the trains' position on a track section. All neural networks of this type are based on the same independently generated dataset containing 100 000 samples. Each network has an input layer with three input neurons and an output layer with two output neurons. The number of hidden layers and neurons per hidden layers varies and is shown in Tab. 1 (*ETCS-**). The inputs are derived from the ETCS-model, this being the max. velocity, the track position train 1, and the track position train 2. Here, the output neurons refer to the two possible advisories for and against braking. In order to evaluate the efficiency of the iSAT solver on deep neural network dimensions, we trained two neural network models based on the MNIST dataset. More exactly, we developed two different variants of the MNIST neural network, both having two hidden layers. The distribution of neurons among the layers is presented in Tab. 1 (*MNIST-**).

Following, we compare the new dedicated sigmoid propagator's runtime and scalability against the composed approach and the approximation approach for each experiment. For all experiments, we used verification targets supposed to be unsatisfiable as well as targets supposed to be satisfiable.

Also, for each experiment, we recorded the following benchmarks: the processor runtime, the solving time, i.e. the time from the beginning of solving (after preprocessing) to the end of the entire process. Furthermore, we recorded the preprocessing time (processor runtime - solving time) in which the given constraint system undergoes a transformation into an internal representation facilitating DPLL-based solving as well as interval constraint propagation, and the number of variables including auxiliary variables introduced by the preprocessing step.

First, we compare the runtime of the new dedicated propagator against the composed approach on the experiments, as can be seen in Fig. 2. Here, the x-axis represents the runtimes of the new dedicated

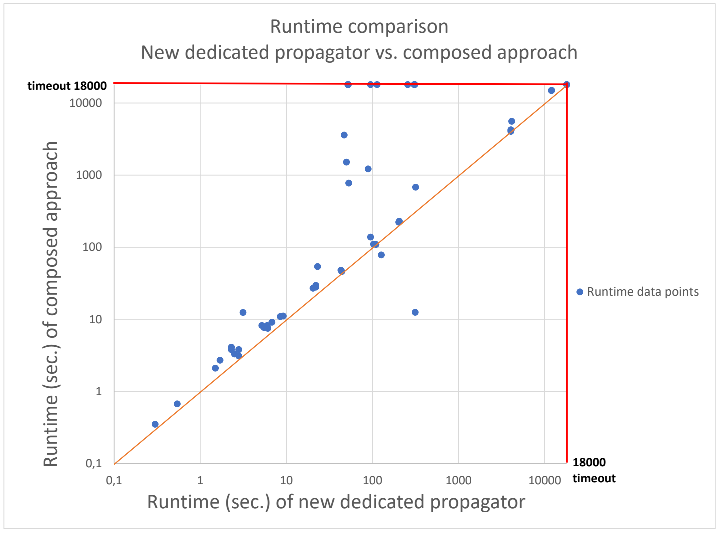

Figure 2: Runtime comparison between the proposed and the composed approach.

sigmoid propagator and the y-axis represents the runtimes of the composed approach, with both axes being scaled logarithmically and where each data point describes a result of an experiment. The evaluation setup sets a process timeout of five hours for each experiment, with the red line marking this limit. The orange diagonal indicates the location of all points for which the runtime of both approaches is identical. As can be observed, in most experiments, the new dedicated propagator has a better runtime. In runtimes where the composed approach had a runtime of more than 500 seconds, the new dedicated sigmoid propagator had a runtime of fewer than 100 seconds. This also includes runtimes in which the composed approach terminated due to a timeout of five hours.

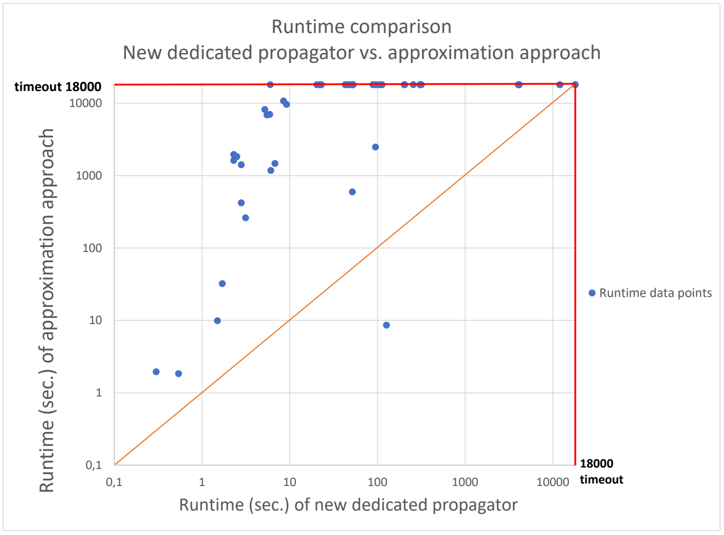

Figure 3: Runtime comparison between the dedicated propagator and approximation approach.

Furthermore, we compare the approximation approach inspired by NeVer against the new dedicated sigmoid propagator. Here, the new dedicated sigmoid propagator's runtime is compared with the runtimes of the approximation approach in Fig. 3. As can be observed, in most experiments, the new dedicated propagator has a better runtime. In contrast to the comparison with the composed approach seen in Fig. 2, the new dedicated sigmoid propagator shows better runtimes even at runtimes below one second. In many cases, the new dedicated sigmoid propagator has a runtime of fewer than 10 seconds, especially where the approximation approach has runtimes from over 1 000 seconds up to a five hour timeout.

Lastly, we evaluate iSAT on the different neural network dimensions presented earlier in Tab. 1 (*ETCS-1* and following), where we consider only the results obtained from the new dedicated propagator. Overall, in our experiments, the solver iSAT requires a longer solving time when more neurons appear in a neural network. Furthermore, for the same number of neurons, the neural network structure can also be a factor when considering the preprocessing time which indicates an apparent reference to an optimizable process in the entire solving process. The preprocessing time depends only on the number of variables in the considered constraint system, while the number of variables are increasing with the number of summations in a neural network model. Thus, in the neural network models with 500 neurons in the hidden layers, the *ETCS-2* neural network model has a significantly higher preprocessing time with about 195 000 variables in contrast to the *ETCS-4* neural network model with about 145 000 variables. In the case of the MNIST neural network models, the input layers consist of 784 neurons and the *ETCS-** models consist of only three input neurons, here, the number of variables being already over 1.1 million in the *MNIST-small* model and over 2.7 million in the *MNIST-big* model. Ultimately, the size of the sum given to the activation function significantly determines the preprocessing time of the iSAT solver. Possible improvement approaches regarding preprocessing are a subject of our current research.

4 Conclusions

We investigated the verification of ANNs with non-linear activation functions using the iSAT solver, with a focus on the sigmoid activation function. For this, we implemented a new dedicated sigmoid propagator into the SMT solver iSAT and evaluated it regarding runtime. The evaluation revealed that the new dedicated sigmoid propagator has better runtime behavior in verifying neural networks than the composition of already implemented propagators. Also, when compared to the approximation approach inspired by NeVer, the new dedicated sigmoid propagator showed a better runtime.

In further investigations, the algorithm of iSAT was analyzed for scalability with respect to the verification of neural networks that make use of the sigmoid function. Here, iSAT achieved a maximum runtime of about 5 minutes for state-of-the-art neural network structures with few hidden layers and up to 500 hidden layer neurons. As stated by the authors in [6], this is a significant increase in the number of neurons that can be handled. Furthermore, according to the approach in [10] of approximating the sigmoid function, 48 hours runtime was already exceeded at a number of 64 hidden neurons. In our experiments with a deep neural network model with 1 176 hidden neurons on the MNIST dataset, iSAT could provide a decision within 3 hours. Despite these positive results, regarding the preprocessing time of the iSAT solver, we discovered an already improvable factor which can speed up the solving process even more. In conclusion, we showed that the algorithm of the SMT solver iSAT is suitable for the verification of large ANNs with non-linear and transcendental functions such as the sigmoid activation function where the iSAT solver benefits from the implementation of a dedicated interval constraint propagator.

References

[1] DB Netz AG (2014): *European Train Control System (ETCS) bei der DB Netz AG*. Technical Report. `https://www.deutschebahn.com/resource/blob/1303328/d9556ec0c860abb53cf07bfcb693f79d/Anhang_Themendienst_ETCS-data.pdf`, last accessed 29.06.2021.

[2] François Chollet: *Keras*. `https://keras.io/`, last accessed 2021-07-08.

[3] Martin Davis, George Logemann & Donald Loveland (1967): *A Machine Program for Theorem-Proving*. Journal of Symbolic Logic 32(1), p. 118, doi:10.2307/2271269.

[4] Martin Fränzle, Christian Herde, Stefan Ratschan, Tobias Schubert & Tino Teige (2007): *Efficient solving of large non-linear arithmetic constraint systems with complex boolean structure*. Journal of Satisfiability, Boolean Modeling and Computation, pp. 209–236, doi:10.3233/SAT190012. Available at `https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.1046.1023&rep=rep1&type=pdf`.

[5] Radoslav Ivanov, James Weimer, Rajeev Alur, George J. Pappas & Insup Lee (2019): *Verisig: Verifying Safety Properties of Hybrid Systems with Neural Network Controllers*. Association for Computing Machinery, New York, NY, USA, doi:10.1145/3302504.3311806. Available at `https://doi.org/10.1145/3302504.3311806`.

[6] Guy Katz, Clark W. Barrett, David L. Dill, Kyle Julian & Mykel J. Kochenderfer (2017): *Reluplex: An Efficient SMT Solver for Verifying Deep Neural Networks*. In Rupak Majumdar & Viktor Kuncak, editors: *Computer Aided Verification - 29th International Conference, CAV 2017, Heidelberg, Germany, July 24-28, 2017, Proceedings, Part I*, *Lecture Notes in Computer Science* 10426, Springer, pp. 97–117, doi:10.1007/978-3-319-63387-9_5. Available at `https://doi.org/10.1007/978-3-319-63387-9_5`.

[7] Yann LeCun, Corinna Cortes & Christopher J. C. Burges: *MNIST*. `http://yann.lecun.com/exdb/mnist/`, last accessed 2021-07-08.

[8] Nina Narodytska, Shiva Prasad Kasiviswanathan, Leonid Ryzhyk, Mooly Sagiv & Toby Walsh (2018): *Verifying Properties of Binarized Deep Neural Networks*. In Sheila A McIlraith & Kilian Q Weinberger, editors: *Proc. of the 32nd AAAI Conf. on AI, (AAAI-18), the 30th innovative Applications of AI (IAAI-18), and the 8th AAAI Symp, on Edu. Advances in AI (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, AAAI Press, pp. 6615–6624. Available at `https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16898`.

[9] Chigozie Nwankpa, Winifred Ijomah, Anthony Gachagan & Stephen Marshall (2018): *Activation Functions: Comparison of trends in Practice and Research for Deep Learning*. Available at `https://arxiv.org/abs/1811.03378`.

[10] Luca Pulina & Armando Tacchella (2010): *An Abstraction-Refinement Approach to Verification of Artificial Neural Networks*. Springer-Verlag, doi:10.1007/$978-3-642-14295-6_24$. Available at `https://doi.org/10.1007/978-3-642-14295-6_24`.

[11] Francesca Rossi, Peter van Beek & Toby Walsh (2006): *Handbook of Constraint Programming (Foundations of Artificial Intelligence)*. Elsevier Science Inc., USA, doi:10.5555/1207782.

[12] AVACS subproject Tool: *Tool iSAT*. Available at `https://projects.avacs.org/projects/isat/`. Last accessed 2021-06-29.

[13] Weiming Xiang, Patrick Musau, Ayana A Wild, Diego Manzanas Lopez, Nathaniel Hamilton, Xiaodong Yang, Joel Rosenfeld & Taylor T Johnson (2018): *Verification for Machine Learning, Autonomy, and Neural Networks Survey*. Available at `https://arxiv.org/pdf/1810.01989.pdf`.