ReCIPH: Relational Coefficients for Input Partitioning Heuristic

Serge Durand, Augustin Lemesle, Zakaria Chihani, Caterina Urban, François Terrier

ReCIPH: Relational Coefficients for Input Partitioning Heuristic

Serge Durand [*1] **Augustin Lemesle** [*1] **Zakaria Chihani** [1] **Caterina Urban** [2] **François Terrier** [1]

Abstract

With the rapidly advancing improvements to the already successful Deep Learning artifacts, Neural Networks (NN) are poised to permeate a growing number of everyday applications, including ones where safety is paramount and, therefore, formal guarantees are a precious commodity. To this end, Formal Methods, a long-standing, mathematically-inspired field of research saw an effervescent outgrowth targeting NN and advancing almost as rapidly as AI itself. Without a doubt, the most challenging problem facing this new research direction is the scalability to the ever-growing NN models. This paper stems from this need and introduces Relational Coefficients for Input partitioning Heuristic (ReCIPH), accelerating NN analysis. Extensive experimentation is supplied to assert the added value to two different solvers handling several models and properties (coming, in part, from two industrial use-cases).

1. Introduction

Arguing for the importance of artificial intelligence (AI) is becoming as superfluous as justifying Computer Science itself. With the resounding successes of the numerous applications, the sheer number of project proposals to generous funding avenues worldwide, the explosion of the number of submissions to the various conferences and workshops in the field, the presence of AI in our everyday lives is a *fait accompli*. Similarly, while less evident, the need for safety in AI is felt throughout the community, with tens of workshops, schools and conferences dedicated to that nascent field (*e.g.,* SafeAI, WAISE, AI Safety, AISecure, DSML, FoMLAS), numerous efforts from certification bodies (*e.g.,* ISO, Afnor) and generous funding (*e.g.,* Trustworthy AI, from the NSF, the French Grand Défi IA de Confiance).

[*]Equal contribution [1] Université Paris-Saclay, CEA, List, F-91120, Palaiseau, France [2]INRIA and Ecole Normale Supérieure, Université PSL Paris, France . Correspondence to: Serge DURAND <serge.durand2@cea.fr>.

1st Workshop on Formal Verification of Machine Learning, Baltimore, Maryland, USA. Colocated with ICML 2022.

What is perhaps still in need of visibility is the role that Formal Methods (FM) in particular can play in this quest for safer ML. Despite historical precedent and a solid track record in the software safety community (Chihani, 2021), there is a consistent belief that AI safety may be beyond what FM can handle. This probably stems from the fact that, while a considerable effort is devoted to the design of frugal AI, most of the ML models remain significantly large, causing a strain on the provers which need to cope with the increase in solving time and the decrease in precision of their analysis. One can address these problems through various means, ranging from parallelisation on multiple servers to new heuristics guiding the solvers to a faster solution. And already the state of the art shows the promise of FM, with numerous tools gaining orders of magnitude in efficiency year after year. For example, one can trace the evolution of Reluplex (Katz et al., 2017) into Marabou(Katz et al., 2019), and the effect of the subsequent improvements (Elboher et al., 2020; Wu et al., 2020b; 2022).

1.1. Our contribution

Our paper aims to participate in this trend by introducing a new heuristic on input partitioning and show its added value to NN verification in the case of low dimensional inputs, which are common in embedded systems where safety is critical such as guidance systems (Julian et al., 2016). Our experimentation section shows that ReCIPH leads to 1 to 2 orders of magnitudes decrease in the number of subproblems to solve compared to other heuristics, with no additional computing cost as it relies on pre-computed linear relaxations. Naturally, this also reduces the solving time.

1.2. Related works

The ReluVal tool (Wang et al., 2018b) relies on input partitioning combined with symbolic interval analysis. The heuristic used to prioritize the partitioning, called the gradient smear score, is based on gradient intervals and requires additional computation, unlike our approach.

In the tool ERAN, linear relaxation is used in the form of abstract domains (DeepPoly (Singh et al., 2019a), and (Singh et al., 2018)). For the ACAS networks (Julian et al., 2016) input partitioning is used as well, using the ReluVal heuris-

tic which is based on gradients. However the partitioning is not a recursive bisection with one input interval split at a time, each interval is split in several intervals depending on its gradient smear score, and a partition of up to thousands intervals is analyzed directly. If necessary additional input partitioning is used to complete the analysis. We believe the justification for such an approach is to minimize the additional cost of computing gradients. However it is not clear how the size of the partition is determined.

(Royo et al., 2019) improves upon Neurify (Wang et al., 2018a) (itself an improvement of ReluVal), using shadow prices, a metric relying on framing the verification problem as a Linear Program (LP). We did not compare our work to it as we could not find an implementation and it is not clear if internal split is used in addition to input partitioning, making comparison with our approach difficult.

Many other works *e.g.,* (Bak et al., 2020; Henriksen & Lomuscio, 2020; Wu et al., 2020a; Wang et al., 2021) use internal splitting, typically by case analysis on the possible status of activation nodes, sometimes in combination with input partitioning, which is particularly well suited for high-dimensional networks but out of this work's scope.

2. Background

2.1. Neural Networks

A neural network f can be considered as a function $f : \mathcal{X} \mapsto \mathcal{Y}$ with $\mathcal{X}$ its input space and $\mathcal{Y}$ its output space. For our work we will focus on feed-forward[1] networks where f is the composition of linear functions f_i and non-linear activation functions σ_i:

$$f(x) = f_l \circ f_{l-1} \circ \sigma_{l-1} \circ \dots \sigma_0 \circ f_0(x)$$

Example of activation functions include:

$$ReLU(x) = max(0, x),$$

$$sig(x) = \frac{1}{1 + e^{-x}},$$

$$tanh(x) = \frac{e^{2x} - 1}{e^{2x} + 1} = 2sig(2x) - 1$$

respectively the Rectified Linear Units (ReLU), the sigmoid and the hyperbolic tangent functions.

NN can be represented as directed graphs with each node (also called neuron) representing the scalar result of the application of preceding linear and activation functions.

[1]In theory our proposed heuristic is compatible with convolutional networks. However convolutional networks are most often applied to high-dimensional inputs, for which input partitioning does not scale.

Let f be a network of l layers, and N nodes. We note $n(k)$ the number of nodes in the the k^{th} layer.

The linear functions are parameterized by matrices of weights W and vectors of biases b of compatible dimensions such that each linear function f_k is formally defined as: $f_k(x) = W_k \cdot x + b_k$, with $W_k \in \mathbb{R}^{n(k) \times n(k-1)}$.

In a node x_i, $i \in \{0, \dots, n(k) - 1\}$ of the k^{th} layer the following computation occurs:

$$x_i = \sigma_k \left(\sum_{j=1}^{n(k-1)} w_{ij} x_j + b_i \right)$$

Our work builds upon linear approximations of such networks, as explained in further sections.

2.2. Neural Network Verification

Given a NN f, a subset $\mathcal{I} \subset \mathcal{X}$ (a specified input region of interest) and a subset $\mathcal{S} \subset Y$ (the safe space) the NN verification problem consist in proving:

$$\forall x \in \mathcal{X}, x \in \mathcal{I} \Rightarrow f(x) \in \mathcal{S}$$

One such problem is ensuring NN robustness to input perturbations, where $\mathcal{I}$ is a ϵ-ball around a given input (a product of intervals in the case of the ℓ_∞ norm) and $\mathcal{S}$ the space where the output corresponds to the ground truth label.

Another classic example is the ACAS Xu family of networks. They consist in 45 NN intended to give advisory to aircraft to avoid collisions. An example of a desirable property is that if two aircrafts are facing each other the score for the advisory "clear of conflict" should not be minimal.[2] More properties are described in (Katz et al., 2017).

The NN verification problem in the case of fully-connected networks with ReLU activation has been proven to be NP-complete (Katz et al., 2017). Methods to tackle this problem include a specialized SMT-solver (Katz et al., 2017), modelisation as an optimization problem (Tjeng et al., 2019), abstract domains tailored-made for the overapproximation of the reachable space $f(\mathcal{I})$ (Singh et al., 2018; 2019a). We refer to (Liu et al., 2019) for details on many methods proposed in the past few years.

2.3. Linear Approximations for Neural Network Verification

The overapproximation of the reachable space is one of the most successful approaches to overcome the scalability problem. The goal is to compute a *conservative* overap-

[2]In the ACAS Xu networks the minimal score correspond the the chosen advisory.

proximation of the reachable space, such that any concrete behaviour is necessarily included in the approximation.

More formally, we note T the transformation that takes a network f and an input space $\mathcal{X}$ as inputs and computes $T(f,\mathcal{I}) \subset \mathcal{Y}$ such that $f(\mathcal{I}) \subset T(f,\mathcal{I})$. Showing that $T(f,\mathcal{I}) \subset \mathcal{S}$ is then sufficient to conclude that for any inputs $x \in \mathcal{I}$ we have $f(x) \in \mathcal{S}$.

The key idea is to approximate the behaviour of the activation functions with a linear approximation. Starting with the inputs, concrete bounds and symbolic relations are computed and propagated in the network through all the layers to finally obtain bounds on each output, which can be used to check whether the property holds.

We explain the idea following a similar formalism as (Singh et al., 2019a). At any given node x_i, $i \in 0, ...N-1$, of the network we have the following information:

$$x_i \leq b^u + \sum_{j=0}^{i-1} \alpha_j^u x_j$$
$$x_i \geq b^l + \sum_{j=0}^{i-1} \alpha_j^l x_j$$
$$l_i \leq x_i \leq u_i$$

where $l_i, u_i \in \mathbb{R}$ are concrete bounds on x_i, $\alpha_j^u, b^u \in \mathbb{R}$ are coefficients representing the upper symbolic relations of x_i with regards to previous nodes, and $\alpha_j^l, b^l \in \mathbb{R}$ the lower symbolic relations.

l_i and u_i can be computed by combining the symbolic relations and previous concrete bounds:

$$l_i = b^l + \sum_{\alpha_j^l \geq 0} \alpha_j^l l_j + \sum_{\alpha_j^l < 0} \alpha_j^l u_j$$

The computation for u_i is analogous. The same or a similar idea is presented in different ways and with different names in other works (*e.g.,* (Wang et al., 2018a), (Wang et al., 2021)). The symbolic coefficients propagation for the linear layers is straightfoward: they are just multiplied with the corresponding weights. For non-linear functions a linear relaxation must be introduced.

Examples of relaxations are represented in Fig 1 and Fig 2. Several choices are possible to have sound approximations, that might imply a different trade-off between precision and performance. Some heuristics choose to have parallel lower and upper symbols by choosing the same coefficient introduced in the linear relaxation and only changing the bias (Wang et al., 2018a), others introduce decoupled coefficients as in (Singh et al., 2019a), and there are works

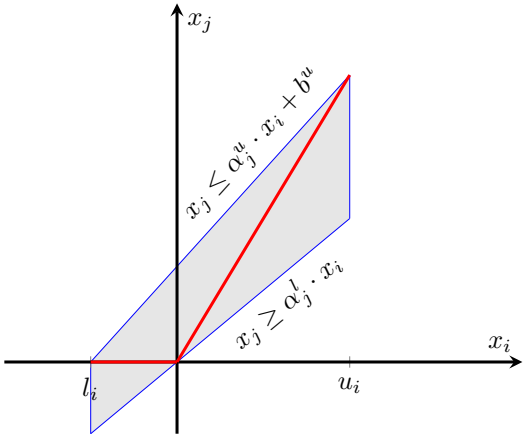

Figure 1. Linear relaxation of the ReLU function

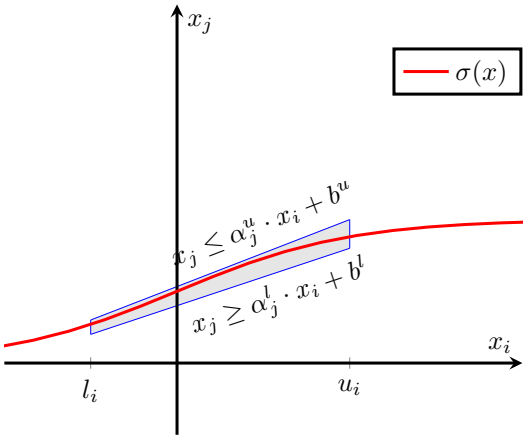

Figure 2. Linear relaxation of the Sigmoid function

trying to find the optimal new coefficient with optimization techniques (Xu et al., 2020). The zonotope domain adapted for NN of (Singh et al., 2018) is presented in a different way but is equivalent to an abstraction with parallel lower and upper symbols. In general the lower and upper bounds of the pre-activation variables always play an important role and help compute the new coefficient used in relaxing the non-linear activations. For the ReLU activations they are used to distinguish between a fixed case (always deactivated or always activated where no relaxation is necessary) and an uncertain case when the input might be positive or negative.

3. Input partitioning

Linear relaxation allows relatively fast analysis of NN but the output bounds are often too imprecise to prove or disprove a property and further refinement is needed. For networks with low-dimensional inputs, a simple but powerful technique to overcome the issue is the partitioning of the input space. The subspaces of the partition are considered as

input spaces of new verification problems to solve. This can be done in a parallelised manner. If the property holds on every subspace of the partition, then the property holds for the original input space. It has been shown in (Wang et al., 2018b) that input partitioning combined with symbolic interval propagation allows for asymptotically complete analysis: it is possible to compute an arbitrarily precise overaprroximation of the reachable space by partitioning the input space in a finite number of subspace.

Minimizing the number of regions to analyze is important: even for relatively small networks it might be necessary to divide the input space into thousands of subspaces. In (Singh et al., 2019b) proving the property 9 of the ACAS Xu benchmark requires a partition into 6 300 regions. We now describe several heuristics to choose the best input to split on during analysis.

3.1. Random choice

A simple heuristic is to choose randomly which input variable to split on. This strategy is used in (Corsi et al., 2020) in a semi-formal framework that estimates output bounds by input sampling. In their work other heuristics do not improve significantly their results. It is also used in Libra (Mazzucato & Urban, 2021), a static analyzer for certifying fairness of NN. We show in 4.3 the gains from using ReCIPH in this case. In our experiments, the random choice does not work well and is significantly worse than the following strategy.

3.2. Biggest interval first

Another simple heuristic is to choose the input with the biggest interval to split on first. Intuitively, the bigger its interval, the more influence the input should have on the outputs. On the upside, this requires no additional computation. However, it does not take into consideration any information from the network or the previous analyses.

3.3. Gradient Smears

In (Wang et al., 2018b) another heuristic to choose the best split is proposed: gradient intervals are computed using the bounds on the outputs and the weights of the network. The final score is computed by using the highest upper bound on the gradient intervals multiplied by the width of each input. This heuristic relies on the intuition that the gradient are a good approximation of the influence of the inputs on the outputs. It uses information from the network and from the output bounds, but it requires additional computation: a backward pass is necessary to compute the gradients.

3.4. ReCIPH

We propose to guide the partitioning using the coefficients computed during analysis with relational domains described in Section 2.3. In particular, let f be a network with n inputs and $N > n$ nodes, and a single output y. For an input $x_i, i \in \{0 \dots n-1\}$, we note L_i the width its input interval.

We consider an analysis with parallel linear approximations for the lower and upper symbolic relations. At the end of a single-pass analysis we obtain:

$$\sum_{i=0}^{N} \alpha_i^u x_i + b^l \leq y \leq \sum_{i=0}^{N} \alpha_i^u x_i + b^u$$
$$l \leq y \leq u$$

To chose the dimension we split on we consider **the magnitude of the coefficients, weighted by the input widths**. The chosen axis is:

$$split\ axis = \max_{i \in \{0 \dots n-1\}} |\alpha_i^u| \times L_i$$

Intuitively, the coefficients approximate the influence of each input on the output. We weight the final coefficients magnitude with the input intervals to take into consideration big disparities between input intervals that could be due to the original specification or repetitive splits on the same interval.

If the network was fully linear, it would be an exact approximation as it would be the combination of all the weights involved in the network. With the non-linearities the new coefficient computed during the linear approximations can be considered as an approximation of the activation function.

In the case of non-parallel linear approximations, with potentially different upper and lower coefficients, we simply choose the axis for which the sum of the absolute values of the coefficients is maximized.

In the case of multiple outputs we have a priority score for each output. The scores can be combined in different manners, depending on the properties we are trying to prove.

Consider a network with three outputs: y_0, y_1, y_2. We use the following combinations:

- If the goal is to prove $y_0 \leq y_1 \wedge y_0 \leq y_2$ we choose the input axis that maximize the scores across outputs.
- If the goal is to prove $y_0 \leq y_1 \vee y_0 \leq y_2$ we choose the input axis that maximize the score across the outputs where we are the closest to finish the proof.
- if the goal is simply to prove $y_0 \geq 0$ (or any other fixed value) we use only the score for y_0.

It is also possible to add layers to have an equivalent network with a single output as described in (Elboher et al., 2019).

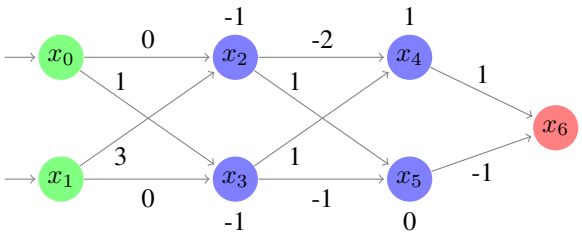

Figure 3. Toy network. Biases are on top and bottom of respective neurons, weights on the edges: for example we have $x_2 = 3 \times x_1 + 0 \times x_0 - 1$ The activation function is a ReLU, only applied after x_2 and x_3. The last layer is added to ease the verification of $x_5 \geq x_4$: it is sufficient to prove $x_6 \leq 0$.

Toy example

We consider the toy network in Fig 3.

We try to prove the following property:

$$\forall x_0, x_1 \in [-2, 2] \times [-0.5, 0.5], x_6 \leq 0$$

A single pass with the DeepPoly abstraction is not sufficient to prove it, yielding:

$$-19.5 \leq x_6 \leq 2.5$$
$$9x_1 - 10 \leq x_6 \leq -1.4x_0 + 11.65x_1 - 5.25$$

We can represent a full analysis as a binary tree as in Fig 4 and Fig 5. Each node represents a single-pass analysis. The red nodes represent nodes where the analysis was not precise enough to prove the property, a split is then necessary to prove the property right or wrong. The label of red nodes are the inputs we bisect. The green nodes represents nodes where the analysis was precise enough to conclude. The children of a node are single-pass analysis on each partition obtained after the bisection of the chosen input interval.

When choosing to split on the widest interval first, we obtain the full analysis represented in Fig 4. Choosing to split according to the ReCIPH scores, we can prove the property in fewer passes (1 split needed and 3 single passes in total, instead of 3 splits and 7 single passes) as illustrated in Fig 5

4. Experimental results

We evaluate the previous heuristics using the Python Reachability Assessment Tool (PyRAT). PyRAT is a static analyzer written in Python and using computing libraries such as NumPy or PyTorch to verify NN properties. Several abstract domains are implemented, including the DeepPoly and DeepZono domains. For the gradient smear heuristic we evaluate it directly using ERAN [3].

[3] https://github.com/eth-sri/eran

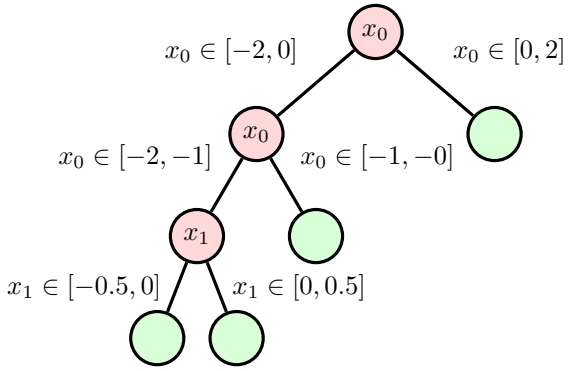

Figure 4. Splitting tree of an analysis of 3 with a split on widest interval heuristic. The labels on edges indicate the size of the new interval after the split.

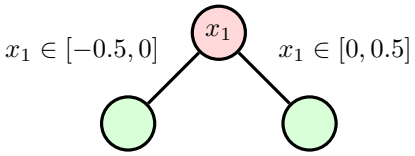

Figure 5. Splitting tree of an analysis of Fig 3 with ReCIPH

4.1. PyRAT on ACAS benchmark

The ACAS benchmark comprises 10 properties (Katz et al., 2017). The first 4 properties are over the 45 networks (Julian et al., 2016) of the benchmark while the other 6 are only defined over one network each.

We present in Table 1 the summarized results on all the networks of property 1, 42 networks of property 3 and 4, and property 5, 9 and 10, analyzed using the DeepPoly abstract domain. They represent a variety of difficulty with some properties verifiable in fewer than 10 passes and other requiring more than 100 000 passes. We notably exclude properties that are violated: most of them requires significantly longer time and we could not run all of them with a reasonable timeout. We did notice similar improvements on a few falsified properties we tested, but overall the input partitioning strategy on its own does not perform well when falsifying properties: an additional search for adversaries is usually used to falsify properties more efficiently.

Compared to splitting on widest input, ReCIPH improves the performance significantly, with up to 19 times fewer analysis required. The time difference is the same: both heuristics do not use additional computation. When comparing ReCIPH to the gradient smear scores, we see that it takes significantly fewer splits on property 3 and 5, respectively 13 and 4 times fewer. For property 4 and 10, the number of splits using ReCIPH is comparable to the use of the gradient smear scores. Nevertheless, even with a similar number of splits, ReCIPH does not require any extra

Table 1. Number of one-pass analysis necessary to prove several properties on the ACAS benchmark, using the DeepPoly domain

Property	# Networks	Width	ReCIPH	Gradient Smear
ϕ_1	45	68467	**15289**	16521
ϕ_3	42	319306	**16140**	209924
ϕ_4	42	25906	**1828**	2206
ϕ_5	1	142637	**7023**	29169
ϕ_9	1	3055	**2395**	3877
ϕ_{10}	1	1331	335	**217**

Table 2. Number of one-pass analysis necessary to prove several properties on a 3x25 fully-connected ReLU network, using the DeepPoly and DeepZono domains (marked respectively with "P" and "Z")

Properties	P Width	P ReCIPH	Z Width	Z ReCIPH
1	2897	**843**	3051	879
2	37917	**6565**	53589	9883
3	6515	**993**	10063	1373
4	9105	**1531**	10203	1751

Table 3. Number of one-pass analysis necessary to prove two properties on a 3x25 fully-connected Sigmoid network, using the DeepZono domain

Property	Width	ReCIPH
1	147	**5**
2*	1446538	**1855**
3*	timeout	**175137**
4	1613	**23**

* Properties proven false.

computation to work and is thus more advantageous. In our experiments, the time required to compute the gradient smear scores takes the majority of the proof's time, more than the passes themselves. For example, it takes 39 seconds out of 51 seconds of the full analysis time for network 1_1 on property 4, which makes more than 75% of the time. Even with the default input partitioning in ERAN with the same property and network, the analysis runs in a total of 28 seconds with 19 seconds for gradient computation. More detailed results are available in Appendix A, showing the time results on properties 1, 3 and 4 and a comparison of the width heuristic and ReCIPH on property 2.

4.2. PyRAT on Mooring lines monitoring NN

We also evaluate our heuristic on an industrial partner use-case which develops a NN for the detection of mooring lines failures on offshore platforms (Sidarta et al., 2018; 2019). The network provided is a small fully-connected network with 3 hidden layers of 25 nodes each and with 7 inputs and 5 outputs. The network exists in two versions, one with ReLU activation functions and the other with sigmoid activation functions. The properties used for this use case are unfortunately under NDA and we cannot detail them. They are domain-specific safety properties in the same spirit as the ACAS properties (Katz et al., 2017). We first analyze the ReLU activation network on four properties using the DeepPoly and the DeepZono domain and detail the results in Table 2. Similarly to the ACAS benchmark, we note a sharp decrease in the number of passes of our analyses with the DeepPoly domain. These results also remain true when using the DeepZono domain, from 3 to 7 times fewer analyses, showing that our heuristic can be extended outside of the DeepPoly domain.

Following this, we test our heuristic on the sigmoid activation network, only using the DeepZono domain on similar properties. As seen in Table 3, ReCIPH also decreases greatly the number of passes required even with a sigmoid activation function and its overapproximation.

4.3. Libra on a fairness benchmark

ReCIPH has also been implemented in Libra (Mazzucato & Urban, 2021), a static analyzer for certifying fairness of fully-connected NN used for classification of tabular data. Libra relies on abstract domains such as DeepPoly and combines a sound forward pre-analysis with an exact backward analysis to quantify the bias in a given input space. In this context, a bias is a part of the input space that leads to a different classification, solely due to a change of value in a sensitive input.

The backward pass complexity is exponential in the number of ReLU nodes with non-fixed status (inputs can be negative or positive). The forward pre-analysis partitions the input space of a ReLU network into *feasible* subspaces by computing a sound overapproximation of the values in each node, and splitting the input space to maximize the number of ReLU nodes with fixed status (either guaranteed to be always active or always inactive). To do the partitioning in a reasonable time, Libra uses an upper bound U on the number of ReLU nodes with non-fixed status, and a lower bound L on the size of the input intervals. The partitioning stops when either the number of ReLU nodes with non-fixed status is below U (the region is considered *feasible* in this case) or if the input intervals get too small, below L, (the region is *unfeasible*, Libra does not proceed with the backward analysis). At the end of the partitioning we thus obtain a coverage of the original input space, the size of which depends on L and U, the abstract domain used, but also the splitting heuristic.

As seen in Table 4 using ReCIPH over the default random strategy for the splitting heuristic of Libra greatly improves the coverage of the feasible space. With the same configuration of L and U it can almost double this feasible space and in all cases the new coverage is above 94%.

Table 4. Percentage of the feasible space, using the DeepPoly domain in Libra for different configuration of L and U

L, U values	Random	ReCIPH
$L = 0.5,\ U = 3$	49.01%	**94.40%**
$L = 0.5,\ U = 5$	56.15%	**97.03%**
$L = 0.25, U = 3$	81.82%	**99.74%**
$L = 0.25, U = 5$	91.58%	**99.98%**

5. Conclusion

In this paper we show that using the coefficients computed during the analysis of NN by linear relaxation, when choosing on which input to split during recursive input partitioning, greatly reduces the number of splits compared to splitting on the biggest input interval first. The gain from ReCIPH over gradient smear (Wang et al., 2018b) used in ERAN is not equally drastic for all the selected properties when only looking at the number of splits. Nevertheless, it is still advantageous *w.r.t.* the analysis time as it does not require extra computation.

Overall, we show that our heuristic is not limited only to one abstract domain, activation function or tool, and that ReCIPH is easily extendable on different linear relaxations and also on different tools with a similar gain.

However, ReCIPH's applicability is currently limited to networks with low-dimensional inputs. While linear relaxation can also be used in convolutional networks specialized in image classification, improvements still need to be made for ReCIPH to make a real impact on their verification. We plan to investigate ReCIPH in this context in the near future, for example to help prioritizing internal split on ReLU nodes.

Acknowledgements

This work was partially funded by the French *Ministère des armées - Agence de l'innovation de défense* (Minstry of Armed Forces - Defence Innovation Agency). This work was also financially supported by the European commission through the SAFAIR subproject of the project SPARTA which has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 830892, as well as through the CPS4EU project that has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 826276. The JU receives support from the European Union's Horizon 2020 research and innovation programme and France, Spain, Hungary, Italy, Germany.

We thank the reviewers and colleagues for their feedback.

References

Bak, S., Tran, H.-D., Hobbs, K., and Johnson, T. T. Improved geometric path enumeration for verifying ReLU neural networks. In *32nd International Conference on Computer-Aided Verification (CAV)*, July 2020. doi: 10.1007/978-3-030-53288-8_4.

Chihani, Z. Formal methods for ai: Lessons from the past, promises of the future. In *CAID 2021: Conference on Artificial Intelligence for Defense*, pp. 62–71, 2021.

Corsi, D., Marchesini, E., and Farinelli, A. Evaluating the safety of deep reinforcement learning models using semi-formal verification. *ArXiv*, abs/2010.09387, 2020.

Elboher, Y. Y., Gottschlich, J., and Katz, G. An abstraction-based framework for neural network verification. *CoRR*, abs/1910.14574, 2019. URL http://arxiv.org/abs/1910.14574.

Elboher, Y. Y., Gottschlich, J., and Katz, G. An abstraction-based framework for neural network verification. In Lahiri, S. K. and Wang, C. (eds.), *Computer Aided Verification*, pp. 43–65, Cham, 2020. Springer International Publishing. ISBN 978-3-030-53288-8.

Henriksen, P. and Lomuscio, A. Efficient neural network verification via adaptive refinement and adversarial search. In *ECAI 2020 proceedings*, 2020.

Julian, K. D., Lopez, J., Brush, J. S., Owen, M. P., and Kochenderfer, M. J. Policy compression for aircraft collision avoidance systems. *2016 IEEE/AIAA 35th Digital Avionics Systems Conference (DASC)*, pp. 1–10, 2016.

Katz, G., Barrett, C., Dill, D. L., Julian, K., and Kochenderfer, M. J. Reluplex: An efficient smt solver for verifying deep neural networks. In Majumdar, R. and Kunčak, V. (eds.), *Computer Aided Verification*, pp. 97–117, Cham, 2017. Springer International Publishing. ISBN 978-3-319-63387-9.

Katz, G., Huang, D. A., Ibeling, D., Julian, K., Lazarus, C., Lim, R., Shah, P., Thakoor, S., Wu, H., Zeljić, A., Dill, D. L., Kochenderfer, M., and Barrett, C. The marabou framework for verification and analysis of deep neural networks. In *To appear in Proceedings of the* 31^{st} *International Conference on Computer Aided Verification (CAV '19)*, July 2019.

Liu, C., Arnon, T., Lazarus, C., Barrett, C., and Kochenderfer, M. J. Algorithms for verifying deep neural networks. *arxiv*, (1903.06758), 2019. URL https://arxiv.org/abs/1903.06758.

Mazzucato, D. and Urban, C. Reduced Products of Abstract Domains for Fairness Certification of Neural Networks. In *28th Static Analysis Symposium (SAS 2021)*, Chicago, United States, October 2021. URL https://hal.inria.fr/hal-03348036.

Royo, V. R., Calandra, R., Stipanovic, D. M., and Tomlin, C. Fast neural network verification via shadow prices. *CoRR*, abs/1902.07247, 2019. URL http://arxiv.org/abs/1902.07247.

Sidarta, D. E., O'Sullivan, J., and Lim, H.-J. Damage detection of offshore platform mooring line using artificial neural network. volume Volume 1: Offshore Technology of *International Conference on Offshore Mechanics and Arctic Engineering*, 06 2018. doi: 10.1115/OMAE2018-77084. URL https://doi.org/10.1115/OMAE2018-77084.

Sidarta, D. E., Lim, H.-J., Kyoung, J., Tcherniguin, N., Lefebvre, T., and O'Sullivan, J. Detection of mooring line failure of a spread-moored fpso: Part 1 — development of an artificial neural network based model. volume Volume 1: Offshore Technology; Offshore Geotechnics of *International Conference on Offshore Mechanics and Arctic Engineering*, 06 2019. doi: 10.1115/OMAE2019-96288. URL https://doi.org/10.1115/OMAE2019-96288. V001T01A042.

Singh, G., Gehr, T., Mirman, M., Püschel, M., and Vechev, M. Fast and effective robustness certification. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 31*, pp. 10802–10813. Curran Associates, Inc., 2018. URL http://papers.nips.cc/paper/8278-fast-and-effective-robustness-certification.pdf.

Singh, G., Gehr, T., Püschel, M., and Vechev, M. An abstract domain for certifying neural networks. *Proceedings of the ACM on Programming Languages*, 3(POPL):1–30, 2019a. doi: 10.1145/3290354. URL https://doi.org/10.1145/3290354.

Singh, G., Gehr, T., Püschel, M., and Vechev, M. Boosting robustness certification of neural networks. In *International Conference on Learning Representations (ICLR)*. 2019b.

Tjeng, V., Xiao, K. Y., and Tedrake, R. Evaluating robustness of neural networks with mixed integer programming. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=HyGIdiRqtm.

Wang, S., Pei, K., Whitehouse, J., Yang, J., and Jana, S. Efficient formal safety analysis of neural networks. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 31*, pp. 6367–6377. Curran Associates, Inc., 2018a. URL http://papers.nips.cc/paper/7873-efficient-formal-safety-analysis-of-neural-networks.pdf.

Wang, S., Pei, K., Whitehouse, J., Yang, J., and Jana, S. Formal security analysis of neural networks using symbolic intervals. In *Proceedings of the 27th USENIX Conference on Security Symposium*, SEC'18, pp. 1599–1614, USA, 2018b. USENIX Association. ISBN 9781931971461.

Wang, S., Zhang, H., Xu, K., Lin, X., Jana, S., Hsieh, C.-J., and Kolter, J. Z. Beta-CROWN: Efficient bound propagation with per-neuron split constraints for neural network robustness verification. In Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=ahYIlRBeCFw.

Wu, H., Ozdemir, A., Zeljic, A., Irfan, A., Julian, K., Gopinath, D., Fouladi, S., Katz, G., Pasareanu, C. S., and Barrett, C. W. Parallelization techniques for verifying neural networks. *CoRR*, abs/2004.08440, 2020a. URL https://arxiv.org/abs/2004.08440.

Wu, H., Ozdemir, A., Zeljić, A., Julian, K., Irfan, A., Gopinath, D., Fouladi, S., Katz, G., Pasareanu, C., and Barrett, C. Parallelization techniques for verifying neural networks. In *2020 Formal Methods in Computer Aided Design (FMCAD)*, pp. 128–137, 2020b. doi: 10.34727/2020/isbn.978-3-85448-042-6_20.

Wu, H., Zeljić, A., Katz, G., and Barrett, C. Efficient neural network analysis with sum-of-infeasibilities. In Fisman, D. and Rosu, G. (eds.), *Tools and Algorithms for the Construction and Analysis of Systems*, pp. 143–163, Cham, 2022. Springer International Publishing. ISBN 978-3-030-99524-9.

Xu, K., Shi, Z., Zhang, H., Wang, Y., Chang, K.-W., Huang, M., Kailkhura, B., Lin, X., and Hsieh, C.-J. Automatic perturbation analysis for scalable certified robustness and beyond. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 1129–1141. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper

/2020/file/0cbc5671ae26f67871cb914d8
1ef8fc1-Paper.pdf.

A. additional results

In Tables 5 6 and 7 we compare the times taken to prove the properties 1, 3 and 4 by ReCIPH and the gradient smear heuristic used in Eran. We use two modes of Eran:

- iterative bisection, where one input is split at a time, as done in PyRAT with ReCIPH or in (Wang et al., 2018b)
- Eran default mode, where the initial partition is done by splitting multiple inputs into multiple partitions each according to their gradient smear, forming a queue of hundreds of subproblems to solve. Those problems are then solved with the iterative bisection. [4]

We run the benchmark on the same machine without parallelisation. We use the DeepPoly domain for both tools. We also exclude the False properties as Eran does not provide falsification with DeepPoly domain without the complete option: the analysis runs until timeout is reached.

As we can see for almost all properties the default Eran fares better than bisection. We believe it is due to two main reasons:

- splitting in a finer partition directly takes less time to compute the gradient
- in some cases it requires fewer subproblems to solve: the ”easy” regions are directly proven safe as the partition is finer, whereas multiple splits might have been necessary with bisection.

In almost all cases ReCIPH performs better. We leave it for future works to investigate adaptive partitioning with ReCIPH. The goal would be to have directly fine-grained partition when the property is deemed hard and bisection if it is deemed easy.

In Table 8 we show that in the case of falsifiable instances ReCIPH outperforms the naive heuristic also with 1-2 order of magnitudes fewer subproblems to solve. Eliminating the safe part of the input space in fewer analysis we can find a region where the property is false faster with ReCIPH.

[4] We did not find the description of the strategy, or the justification for the choices of the size of the partition, in the literature. We describe here what we could gather from Eran’s code.

Table 5. Full time results on ACAS Property 1, in seconds.

Network	Eran Bisection	Eran Default	PyRAT ReCIPH
1_1	5.40	0.66	0.45
1_2	5.41	0.54	0.49
1_3	8.03	1.23	1.00
1_4	4.75	1.13	0.63
1_5	3.87	0.62	0.34
1_6	3.97	0.59	0.50
1_7	2.43	0.62	0.31
1_8	1.33	0.57	0.18
1_9	0.93	0.68	0.14
2_1	8.05	0.64	0.69
2_2	20.51	3.07	1.47
2_3	9.64	0.88	1.06
2_4	7.39	0.66	0.78
2_5	28.22	10.26	2.12
2_6	18.85	5.95	2.08
2_7	58.71	18.92	5.16
2_8	65.78	16.51	4.99
2_9	73.26	20.52	7.08
3_1	19.82	1.89	0.86
3_2	10.25	2.47	0.87
3_3	11.47	1.01	0.86
3_4	8.35	0.80	0.61
3_5	25.54	3.94	1.45
3_6	48.54	13.52	3.89
3_7	52.40	14.83	4.29
3_8	54.56	13.09	3.55
3_9	78.94	23.30	5.80
4_1	33.12	6.56	1.85
4_2	15.50	1.13	1.18
4_3	11.31	0.96	0.86
4_4	8.58	1.12	0.82
4_5	21.97	4.36	1.94
4_6	71.87	27.81	6.29
4_7	200.80	49.14	10.21
4_8	38.64	16.70	4.57
4_9	135.64	42.52	9.06
5_1	9.70	0.86	0.81
5_2	10.55	1.31	0.84
5_3	5.62	0.52	0.66
5_4	5.65	0.78	0.62
5_5	15.19	4.55	1.50
5_6	43.15	8.96	3.41
5_7	39.56	8.55	3.40
5_8	61.39	21.59	6.69
5_9	46.44	13.89	5.54
total	1411.07	370.21	111.93

Table 6. Full time results on ACAS Property 3, in seconds.

Network	Eran Bisection	Eran Default	PyRAT ReCIPH
1_1	timeout (> 180)	timeout (> 180)	103.27
1_2	39.59	17.42	2.39
1_3	timeout (> 180)	44.41	4.71
1_4	1.58	1.49	0.07
1_5	1.51	1.10	0.17
1_6	0.08	0.01	0.01
2_1	11.68	3.87	1.28
2_2	15.76	2.90	1.90
2_3	12.12	1.67	1.25
2_4	0.07	0.01	0.01
2_5	0.34	0.58	0.04
2_6	0.14	0.01	0.01
2_7	0.21	0.01	0.01
2_8	0.46	0.01	0.01
2_9	0.01	0.01	0.01
3_1	1.57	0.95	0.09
3_2	46.06	1.16	0.44
3_3	3.52	1.31	0.35
3_4	0.92	1.04	0.09
3_5	0.21	1.03	0.02
3_6	0.73	1.00	0.09
3_7	0.08	0.01	0.01
3_8	0.46	0.88	0.02
3_9	0.53	1.25	0.07
4_1	20.31	2.70	0.88
4_2	40.06	17.10	2.20
4_3	23.34	1.60	0.96
4_4	0.26	0.57	0.04
4_5	0.20	0.01	0.01
4_6	1.57	0.92	0.07
4_7	0.59	1.00	0.04
4_8	0.27	0.01	0.01
4_9	0.53	1.01	0.06
5_1	43.44	28.10	2.35
5_2	2.48	4.11	0.34
5_3	1.03	1.10	0.09
5_4	0.34	1.14	0.02
5_5	0.66	1.02	0.06
5_6	0.59	1.37	0.06
5_7	0.01	0.01	0.01
5_8	0.97	1.10	0.06
5_9	0.52	0.01	0.01
total*	274.84	100.59	15.54

*Excluding networks 1_1 and 1_3 because of timeouts

Table 7. Full time results on ACAS Property 4, in seconds.

Network	Eran Bisection	Eran Default	PyRAT ReCIPH
1_1	61.64	26.16	11.12
1_2	16.24	4.83	1.86
1_3	11.57	3.24	1.57
1_4	3.61	1.43	0.15
1_5	4.19	1.30	0.21
1_6	2.14	1.46	0.16
2_1	1.44	1.19	0.11
2_2	0.95	1.47	0.11
2_3	0.39	1.94	0.09
2_4	0.21	0.98	0.02
2_5	1.06	0.93	0.06
2_6	0.41	1.05	0.06
2_7	0.34	1.60	0.02
2_8	1.13	1.39	0.07
2_9	0.08	0.01	0.01
3_1	0.72	2.06	0.18
3_2	0.68	1.45	0.10
3_3	0.65	0.01	0.01
3_4	0.21	1.00	0.03
3_5	0.75	1.05	0.06
3_6	0.21	1.44	0.02
3_7	0.23	1.07	0.03
3_8	0.58	1.14	0.08
3_9	0.63	1.20	0.06
4_1	0.08	0.01	0.01
4_2	0.90	1.03	0.06
4_3	1.38	1.45	0.12
4_4	0.52	1.05	0.06
4_5	0.15	1.37	0.03
4_6	0.22	1.74	0.03
4_7	0.35	1.41	0.06
4_8	1.50	1.46	0.04
4_9	0.30	1.56	0.02
5_1	0.35	2.11	0.03
5_2	0.14	3.38	0.03
5_3	0.21	1.28	0.06
5_4	0.27	0.99	0.03
5_5	0.54	1.04	0.02
5_6	0.25	0.01	0.01
5_7	0.21	0.01	0.01
5_8	0.48	1.24	0.04
5_9	0.15	1.90	0.02
total	118.06	82.41	16.86

Table 8. Number of passes required to verify / falsify instances of ACAS property 2 with the Zonotope domain

	Width		ReCIPH	
Network	result	time	result	time
1_1	True	31229	True	1071
1_2	False	138919	False	6540
1_3	False	100105	False	36833
1_4	False	71049	False	23556
1_5	Timeout	556689	False	314555
1_6	False	312504	False	22636
1_7	True	7463	True	949
1_8	True	15657	True	1295
1_9	True	11239	True	891
2_1	False	97321	False	3931
2_2	False	125611	False	16229
2_3	False	13552	False	1940
2_4	False	84291	False	5855
2_5	False	57268	False	6758
2_6	False	131029	False	7572
2_7	False	23179	False	6127
2_8	False	11204	False	3200
2_9	False	148025	False	10621
3_1	False	126230	False	2768
3_2	Timeout	544751	False	70850
3_3	Timeout	538809	True	166557
3_4	False	163273	False	7853
3_5	False	137982	False	5425
3_6	False	23754	False	6212
3_7	Timeout	541465	False	81287
3_8	False	20076	False	4091
3_9	False	22377	False	2521
4_1	False	103739	False	8317
4_2	Timeout	532931	True	392593
4_3	False	89251	False	3119
4_4	False	106581	False	6574
4_5	False	110367	False	7626
4_6	False	72296	False	9265
4_7	False	164381	False	8475
4_8	False	26609	False	6627
4_9	Timeout	531291	False	19146
5_1	False	89479	False	2561
5_2	False	87557	False	3206
5_3	Timeout	534549	Timeout	571225
5_4	False	82415	False	4293
5_5	False	104168	False	1442
5_6	False	78806	False	5444
5_7	False	13703	False	1868
5_8	False	10580	False	2829
5_9	False	17810	False	6759
total (w-o timeouts)	-	3031079	-	263279