Ecole Polytechnique
INF575. Safe Intelligent Systems

Project. Verification of neural networks

Recent years have shown a broad adoption of deep neural networks in safety-critical applications, typically in autonomous vehicles. A fundamental challenge remains: to ensure the safety of these learning enabled systems.

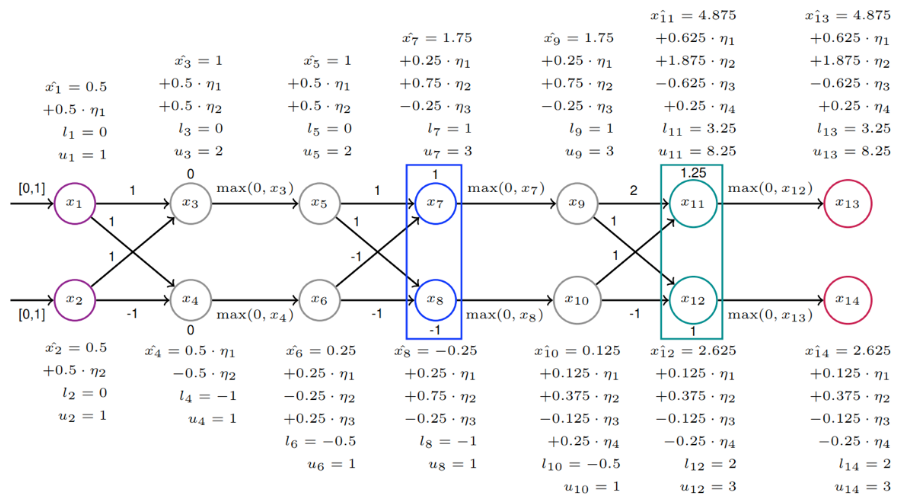
In particular, neural networks are often vulnerable to small disturbances, which may lead to incorrect decisions. To address this challenge and prove that a network is free of adversarial examples (usually, in a region around a given input), recent work has started investigating the use of verification. Current verifiers can be broadly classified as either complete or incomplete. Complete verifiers are exact, i.e., if the verifier fails to certify a network then the network is non-robust (and vice-versa). They are mostly based on Mixed Integer Linear Programming (MILP) such as e.g. [4] or SMT solvers [9]. Although precise, these can only handle networks with a small number of layers and neurons. To scale, incomplete verifiers usually employ overapproximation methods and hence they are sound but may fail to prove robustness even if it holds. In this second class of methods, an approach has been recently proposed [5], allowing to prove by abstract interpretation, the robustness to disturbances for neural networks of realistic size. Variations were then proposed to improve the tradeoff between accuracy and speed [12, 13]. These approaches are based on the abstract domain of zonotopes [6], with different variations for the abstraction of the activation functions, all aiming at scaling-up verification for feedforward neural networks.

The present project aims at proposing, implementing and experimenting an alternative refinement of the zonotopic abstraction in the case of the RELU activation function, using the interpretation of conditionals proposed in cite [7]. The aim being to outperform the existing refinement [13] in speed, while keeping a similar precision. (although for the course evaluation, a project may be considered successful even without completely reaching this ambitious aim...) A complementary and related goal is to extend these promising analyzes to broader classes of neural networks, such as recurrent networks, or more general safety properties such as global robustness.

Of course, you are very welcome to propose other (possibly non zonotopic) abstractions, but you will be expected to explain why you think they may be well suited to neural networks, and demonstrate/compare their behavior with respect to baselines.

1 Abstraction

In [12], an abstraction is proposed to over-approximate the behavior of the activation functions such as the RELU function $y = \max(0, x)$, which leads for example to the results of Figure 1. This abstraction, while being already an improvement over the initial zonotopic abstraction proposed in [5], is still imprecise, as pointed out by the example of Figure 1. In [13], a refinement of the abstraction of [12] is proposed, by combining it to MILP (Mixed Integer Linear Programming) and LP (Linear Programming) relaxations. On this example, for example, the method of [13] will allow to prove that x_{13} is greater than x_{14}, which is not the case for the abstraction of [12].



$\hat{x_1} = 0.5$ $+0.5 \cdot \eta_1$ $l_1 = 0$ $u_1 = 1$
$\hat{x_2} = 0.5$ $+0.5 \cdot \eta_2$ $l_2 = 0$ $u_2 = 1$
$\hat{x_3} = 1$ $+0.5 \cdot \eta_1$ $+0.5 \cdot \eta_2$ $l_3 = 0$ $u_3 = 2$
$\hat{x_4} = 0.5 \cdot \eta_1$ $-0.5 \cdot \eta_2$ $l_4 = -1$ $u_4 = 1$
$\hat{x_5} = 1$ $+0.5 \cdot \eta_1$ $+0.5 \cdot \eta_2$ $l_5 = 0$ $u_5 = 2$
$\hat{x_6} = 0.25$ $+0.25 \cdot \eta_1$ $-0.25 \cdot \eta_2$ $+0.25 \cdot \eta_3$ $l_6 = -0.5$ $u_6 = 1$
$\hat{x_7} = 1.75$ $+0.25 \cdot \eta_1$ $+0.75 \cdot \eta_2$ $-0.25 \cdot \eta_3$ $l_7 = 1$ $u_7 = 3$
$\hat{x_8} = -0.25$ $+0.25 \cdot \eta_1$ $+0.75 \cdot \eta_2$ $-0.25 \cdot \eta_3$ $l_8 = -1$ $u_8 = 1$
$\hat{x_9} = 1.75$ $+0.25 \cdot \eta_1$ $+0.75 \cdot \eta_2$ $-0.25 \cdot \eta_3$ $l_9 = 1$ $u_9 = 3$
$\hat{x_{10}} = 0.125$ $+0.125 \cdot \eta_1$ $+0.375 \cdot \eta_2$ $-0.125 \cdot \eta_3$ $+0.25 \cdot \eta_4$ $l_{10} = -0.5$ $u_{10} = 1$
$\hat{x_{11}} = 4.875$ $+0.625 \cdot \eta_1$ $+1.875 \cdot \eta_2$ $-0.625 \cdot \eta_3$ $+0.25 \cdot \eta_4$ $l_{11} = 3.25$ $u_{11} = 8.25$
$\hat{x_{12}} = 2.625$ $+0.125 \cdot \eta_1$ $+0.375 \cdot \eta_2$ $-0.125 \cdot \eta_3$ $-0.25 \cdot \eta_4$ $l_{12} = 2$ $u_{12} = 3$
$\hat{x_{13}} = 4.875$ $+0.625 \cdot \eta_1$ $+1.875 \cdot \eta_2$ $-0.625 \cdot \eta_3$ $+0.25 \cdot \eta_4$ $l_{13} = 3.25$ $u_{13} = 8.25$
$\hat{x_{14}} = 2.625$ $+0.125 \cdot \eta_1$ $+0.375 \cdot \eta_2$ $-0.125 \cdot \eta_3$ $-0.25 \cdot \eta_4$ $l_{14} = 2$ $u_{14} = 3$
[0,1]
[0,1]
$\max(0, x_3)$
$\max(0, x_4)$
$\max(0, x_7)$
$\max(0, x_8)$
$\max(0, x_{12})$
$\max(0, x_{13})$
1.25

Figure 1: Example and analysis with DeepZ (see [13] for details)

In this project, I suggest that you design, implement and experiment an alternative to this refinement. Before designing your own abstraction, you should:

- understand the context and aim of such analyzes [5]
- make sure you understand the existing abstractions, and in particular are able to reproduce results of the example of Figure 1, obtained with the abstraction of [12] and their improvement in [13].

1.1 An alternative for the abstraction of the RELU activation function $y = \max(0, x)$

Let us first remark that $y = \max(0, x)$ can be written as a conditional statement:

```
if (x >= 0)
  then y = x;
else
  y = 0;
```

Based on this remark, I suggest to use for the abstraction of RELU, the abstraction of conditional statements of [7], where noise symbols ε_i no longer lie in [-1,1], but can be constrained, leading to a refinement of zonotopes, as hinted in next paragraph.

Abstraction of conditional statements There is no natural intersection on zonotopes. We proposed in [7] an approach to handle these intersections, introducing an abstract domain on the noise symbols to take into account the constraints induced by tests. In order to give a short intuition on that approach, let us consider the following program:

```
x = [0 ,1];      // (1) x = 0.5 + 0.5 eps1
y = 2*x;         // (2) y = 1 + eps1 in [0 ,2]
if (y >= 1)         (3)
   z = y -x;        (4)
```

Interpreting this example using a (classical) reduced product of affine sets with boxes (or intervals), we have at control point (3) $y = 1 + \varepsilon_1 \cap [1, 2]$ and $x = 0.5+0.5\varepsilon_1$. We deduce for example at control point (4) $z = 0.5+0.5\varepsilon_1 \cap([1, 2]-[0, 1]) \in [0, 1]$, which is imprecise. Indeed, the constraint on y does not act on the value of x.

Now, instead of considering the noise symbols to be always in $[-1, 1]$, we can interpret the constraint y ¿= 1 as a constraint on noise symbols, here $y = 1 + \varepsilon_1 \geq 1$, which results in $\varepsilon_1 \geq 0$. Here, this constraint can be exactly abstracted in intervals, by $\varepsilon_1 \in [0, 1]$. At point (3) we now have $y = 1 + \varepsilon_1$ and $x = 0.5 + 0.5\varepsilon_1$, with $\varepsilon_1 \in [0, 1]$, at point (4) $z = 0.5 + 0.5\varepsilon_1$ with $\varepsilon_1 \in [0, 1]$, and thus $z \in [0.5, 1]$. The idea when constraining the noise symbols is that we consider only the subset of possible executions for which input x is positive.

In the general case of more general constraints, where several noise symbols may be involved, abstraction of these constraints in another abstract domain, such as zones, octagons, etc, may be useful to refine over simple constraint propagation in intervals.

In the context of this project In this project, the main point will be working on the choice of abstractions and heuristics to (soundly) abstract these constraints on the noise symbols. Indeed, additional constraints and cases disjunctions will be possibly introduced whenever a RELU node is encountered, which may a lot in a large network. A baseline can be a version where all constraints are kept exactly, no join are computed, and a LP solver is used when evaluation is needed. Of course, many disjunctions will appear and the approach will likely be very costly.

Then, heuristics can be explored, in order to avoid too many disjunctions of constraints while keeping good accuracy for large networks. In particular you may try to

- simplify sets of constraints when possible,

- join abstract values (affine forms taken with their constraints) that are ”close enough” in the disjunction.

Some distance measure thus has to be designed, and based on this, a join between well chosen close enough values will be computed, to reduce the number of disjunctions while keeping a good precision. Abstractions for the valeus taken by the noise symbols will probably be useful.

I suggest to first validate your ideas manually on the example of Figure 1, before getting into too much implementation...

2 Implementation and experimentation

2.1 Implementation

You can try to develop your abstraction based on ERAN (`https://github.com/eth-sri/eran`) which relies on the ELINA (`http://elina.ethz.ch`) library of abstract domains, and was the basis for the experiments of [5, 12, 13]. Ideally, you would create your own abstract domain in ELINA. As we want to propose an alternative to DeepZ and RefineZono, this would be the most convenient and satisfying way to provide serious performances comparisons, and try new examples. However, it is not clear how much time will be needed to get into the existing code. Note that ERAN/ELINA also include a polyhedric abstraction DeepPoly [14], but we will not consider it here.

Alternatively, you can re-develop everything by yourself (but you would then either need to redevelop a parser or be limited to small examples). In this case, you can imagine using a parser for simpler (but less general/classical) formats for representing neural networks, which was the choice when designing the sherlock [2, 4] tool for range analysis of neural networks (`https://github.com/souradeep-111/sherlock`, see `https://github.com/souradeep-111/sherlock/blob/master/sherlock-network-format.pdf` for the formal). Naturally, comparisons to baseline will be more tedious.

2.2 Validation

As already mentioned, the toy example of Figure 1 is a good first test for your abstraction. I expect you to present the results (and details of the computation) of your analyzes on this example.

Then, if possible, present some experimental results on the benchmarks of the baselines.

3 Going further

Some hints for further work include the following

3.1 Recurrent networks

The networks currently considered are feedforward and convolutional networks. A direction for further work would be to consider simple recurrent networks. A fixpoint computation will a priori be needed, that can be inspired by fixpoint computation in program analysis. I believe that an abstraction of RELU function with abstraction of constraints on noise symbols should be reasonably well suited to such fixpoint computation.

3.2 Abstractions for other activation functions?

3.3 Analysis of the full closed-loop system

A further step is to prove safety and/or stability of full (continuous/hybrid) closed-loop systems with neural network components, for example a neural network controller for a plant model described as a continuous system, as represented Figure 2.

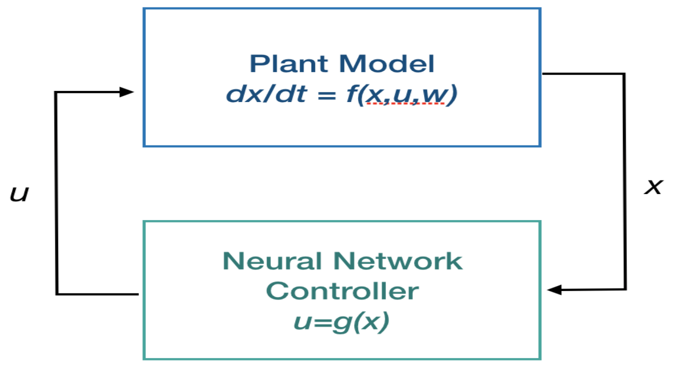


Figure 2: Closed-loop system with neural network component

For this, an analysis such as above will have to be coupled with some reachability analysis for the continuous part, such as developed in the other verification project. You can join force with a team/project working on reachability analysis You can draw benchmark examples from [11] for example, or work also on modeling/building your own simple neural network controller. An alternative is to team up with a third team/project who would focus on building and training a neural network controller for a realistic application, such as for example with reinforcement learning for UAV control [10] or using model predictive control and counter-examples to safety properties to train the network [1] (probably more risky for this 3rd team...). In both cases, it ultimately should lead to an alternative to existing work on the verification of neural network controllers, among which [3, 8]. Some recent work [15] also aims at verifying the full end-to-end verification problem, where the neural network controller processes LIDAR images to produce control actions on the autonomous robot. Trying to model and address a similar problem (avoiding some of the partitioning?) would be interesting of course.

References

[1] Arthur Clavière, Souradeep Dutta, and Sriram Sankaranarayanan. Trajectory tracking control for robotic vehicles using counterexample guided training of neural networks. In *Proceedings of the Twenty-Ninth International Conference on Automated Planning and Scheduling, ICAPS 2018, Berkeley, CA, USA, July 11-15, 2019.*, 2019. URL: `https://aaai.org/ojs/index.php/ICAPS/article/view/3555`.

[2] Souradeep Dutta, Xin Chen, Susmit Jha, Sriram Sankaranarayanan, and Ashish Tiwari. Sherlock - a tool for verification of neural network feedback systems. In *Proceedings of the 22Nd ACM International Conference on Hybrid Systems: Computation and Control*, HSCC '19, 2019.

[3] Souradeep Dutta, Xin Chen, and Sriram Sankaranarayanan. Reachability analysis for neural feedback systems using regressive polynomial rule inference. In *Proceedings of the 22Nd ACM International Conference on Hybrid Systems: Computation and Control*, HSCC '19, 2019.

[4] Souradeep Dutta, Susmit Jha, Sriram Sankaranarayanan, and Ashish Tiwari. Output range analysis for deep feedforward neural networks. In *NASA Formal Methods - 10th International Symposium, NFM 2018*, 2018.

[5] T. Gehr, M. Mirman, D. Drachsler-Cohen, P. Tsankov, S. Chaudhuri, and M. Vechev. Ai2: Safety and robustness certification of neural networks with abstract interpretation. In *2018 IEEE Symposium on Security and Privacy (SP)*, 2018.

[6] Khalil Ghorbal, Eric Goubault, and Sylvie Putot. The zonotope abstract domain taylor1+. In *Proceedings of the 21st International Conference on Computer Aided Verification*, CAV '09, 2009.

[7] Khalil Ghorbal, Eric Goubault, and Sylvie Putot. A logical product approach to zonotope intersection. In *Proceedings of the 22Nd International Conference on Computer Aided Verification*, CAV'10, 2010.

[8] Radoslav Ivanov, James Weimer, Rajeev Alur, George J. Pappas, and Insup Lee. Verisig: Verifying safety properties of hybrid systems with neural network controllers. In *Proceedings of the 22Nd ACM International Conference on Hybrid Systems: Computation and Control*, HSCC '19, 2019.

[9] Guy Katz, Clark Barrett, David Dill, Kyle Julian, and Mykel Kochenderfer. Reluplex: An efficient smt solver for verifying deep neural networks, 2017.

[10] William Koch, Renato Mancuso, Richard West, and Azer Bestavros. Reinforcement learning for uav attitude control. *ACM Trans. Cyber-Phys. Syst.*, 2019. URL: `http://doi.acm.org/10.1145/3301273`.

[11] Diego Manzanas Lopez, Patrick Musau, Hoang-Dung Tran, and Taylor T. Johnson. Verification of closed-loop systems with neural network controllers. In *ARCH19. 6th International Workshop on Applied Verification of Continuous and Hybrid Systems*. URL: `https://easychair.org/publications/paper/ZmnC`.

[12] Gagandeep Singh, Timon Gehr, Matthew Mirman, Markus Puschel, and Martin Vechev. Fast and effective robustness certification. In *Advances in Neural Information Processing Systems 31*. 2018.

[13] Gagandeep Singh, Timon Gehr, Markus Puschel, and Martin Vechev. Boosting robustness certification of neural networks. In *International Conference on Learning Representations*. URL: `https://files.sri.inf.ethz.ch/website/papers/RefineZono.pdf`.

[14] Gagandeep Singh, Timon Gehr, Markus Püschel, and Martin Vechev. An abstract domain for certifying neural networks. *POPL*, 2019.

[15] Xiaowu Sun, Haitham Khedr, and Yasser Shoukry. Formal verification of neural network controlled autonomous systems. In *Proceedings of the 22Nd ACM International Conference on Hybrid Systems: Computation and Control*, HSCC '19, 2019. URL: `https://arxiv.org/abs/1810.13072`.