

WEST UNIVERSITY OF TIMIȘOARA
FACULTY OF MATHEMATICS AND COMPUTER
SCIENCE
MASTER STUDY PROGRAM: SOFTWARE
ENGINEERING

Verification of ACAS-Xu benchmark using Alpha Beta Crown and Nnenum

Supervisor:
Mădălina Erașcu

Students:
Amalia Duma Diana
Răzvan Maciovan Alexandru
Luca Mitroi
Vlad Petcu
Vlad Țîrcomnicu

TIMIȘOARA
2024

1 Introduction

The assignment for this project, required the usage of one benchmark and two tools for which to analyze upon. For the purpose of our project, we have decided to utilize the ACAS-Xu benchmark¹, along with the tools Alpha Beta Crown² and Nnenum³. In the following sections we will see what the selected benchmark represents (section 2), the challenges the tools presented (section 3), as well as how ACAS-Xu scored with the chosen tools based on the VNN COMP 2023 competition (section 4).

2 ACAS-Xu Dataset

ACAS stands for Airborne Collision Avoidance System. There are multiple types of ACAS benchmarks, but the one this paper is based on is Xu, which is optimized for unmanned aircraft systems (UAS), issuing turn rate advisories to remote pilots[3]. The installation process for the benchmark was easy, only needing to download it from GitHub¹.

The input data that can be found across the VNN-LIB files, revolves around different aircraft parameters determined from sensor measurements[5]. These parameters can be seen in figure 1:

- ρ - Distance from ownership to intruder;
- θ - Angle to intruder relative to aircraft heading direction length;
- ψ - Heading angle of intruder relative to aircraft heading direction;
- v_{own} - Speed of aircraft;
- v_{int} - Speed of intruder;
- τ - Time until loss of vertical separation;
- a_{prev} - Previous advisory;

These values were introduced into the neural network (the ONNX files), representing a combination of 45 deep neural networks. The choice of having 45 DNNs comes from the need to reduce the lookup time, as previously there was only one DNN with 2GB in size [3]. They were created by mixing the time until loss of vertical separation (τ) and the previous advisory (a_{prev}). Therefor every network has five inputs and five outputs. The inputs are the aforementioned ones (excluding τ and a_{prev}), while the outputs represent the different horizontal advisories provided to the aircraft: Clear-of-Conflict (COC), weak right, strong right, weak left, or strong left. The DNNs are fully connected, use ReLU activation functions, and have six hidden layers with a total of 300 ReLU nodes each. [3].

In order to confirm the correctness of the results, figure 2 illustrates a top-down of a head-on encounter scenario, where each pixel shows a different course of action based on the presence of an intruder.

¹https://github.com/ChristopherBrix/vnncomp2023_benchmarks/tree/main/benchmarks/acasxu

²<https://github.com/Verified-Intelligence/alpha-beta-CROWN>

³<https://github.com/stanleybak/nnenum>

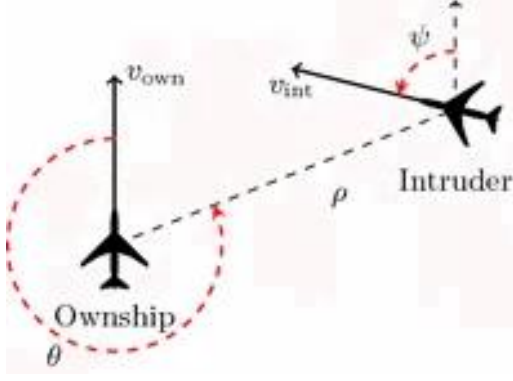


Figure 1: Geometry of ACAS Xu horizontal logic table [3]

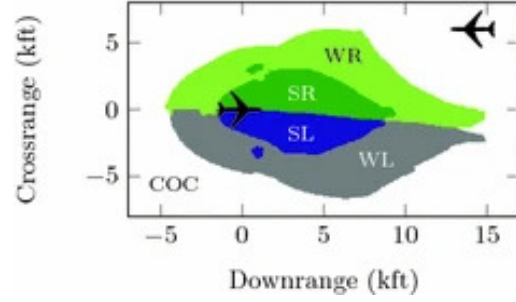


Figure 2: Advisories for a head-on encounter with $a_{\text{prev}} = \text{COC}$ and $\tau = \text{os}$ [3]

3 Tools

For this verification assignment, two tools were used to ensure the reliability and safety of ACAS XU: alpha-beta-CROWN and nnenum. Alpha Beta Crown serves as a neural network verifier based on an efficient linear bound propagation framework, built on bound-propagation-based neural network verifiers: CROWN [8, 11], alpha-CROWN [9], beta-CROWN [7], GCP-CROWN [10], and nonlinear branch-and-bound [6]. Nnenum is a verifier that relies on multiple levels of abstraction to achieve high performance verification of ReLU networks without sacrificing completeness [1]. In the following sections, the setup of these tools is presented.

3.1 Alpha-Beta-CROWN

Setting it up involved a two-step process: first, installing Miniconda and then the tool itself. There were no complications in the first part, as there is an executable file available on their official site that takes care of the entire setup once run. However, the latter part, involving the tool’s installation, presented numerous complications.

The setup of Alpha Beta Crown followed their "Installation and Setup" section outlined on GitHub. The first command was supposed to clone their GitHub repository including "auto_LiRPA" [8], an essential component for the tool’s functionality. Despite running this command, only the tool’s files were retrieved due to permission denials that restricted access to the second GitHub repository. Resolving this required an additional step: manually obtaining the missing necessary files and placing them in their indicated location. The next phase involved creating the conda environment using the specified command, which led to an error that can be observed in figure 3.

This error consumed a considerable amount of time, as the error message initially suggested an issue with the script. After extensive investigation, the root cause was traced to the configuration file. Instead of containing the necessary configuration for the conda environment, it referenced another configuration file by name. After resolving the configuration file issue, the installation of the tool seemed successful. However, when attempting to run it, another problem emerged indicating a missing library, "auto_LiRPA". Although the necessary files existed, the error message

side value. This particular value is used to assess whether the property is satisfied or not, since a negative value indicates a potential violation of the property. One example of such iteration can be observed in figure 5.

```
Iteration 1
Batch size: 1000
Worst bound: tensor([-421.77920532], device='cuda:0')
Total time: 2.1163 pickout: 0.0006 decision: 0.0187 bounding: 2.0860 add_domain: 0.0110
length of domains: 32
32 branch and bound domains visited
Current (lb-rhs): -421.7792053222656
Cumulative time: 6.288111686706543
```

Figure 5: Alpha-Beta-CROWN first iteration in the verification process

In contrast to alpha-beta-CROWN, nnenum differs in that it produces more concise output, focusing primarily on the verification result and essential statistics such as runtime, work fraction completed, and the count of symbolic abstract states ('stars') evaluated. In the case of an unsafe result, a counterexample that demonstrates the violation is also shown. One such output is presented in figure 6.

```
Running net 2-9 with spec 8
Running in parallel with 12 processes
(0.0 sec) Q: 0, Sets: 0/1 (0.0%) ETA: - (expected 1 stars)
(0.2 sec) Q: 4, Sets: 6/65 (0.172%) ETA: 1.56 min (expected 3495 stars)
Found unsafe from second concrete execution of abstract counterexample

violation star found a confirmed counterexample.

Unsafe Base Branch: +++-+++ (Mode: 0)
(0.3 sec) Q: 0, Sets: 63/152 (1.126%) ETA: 23.4 sec (expected 5595 stars)

Total Stars: 63 (1 exact, 62 approx)
Runtime: 0.3 sec
Completed work frac: 0.011260032653808594
Num Stars Copied Between Processes: 17
Num Lps During Enumeration: 15
Total Num Lps: 15

Result: network is UNSAFE with confirmed counterexample in result.cinput and result.coutput
Input: [-0.22267237305641174, -0.49999991059303284, -0.012686670757830143, 0.31314754486083984, 0.2928498089313507]
Output: [0.05460606515407562, -0.02080647647380829, 0.024885952472686768, -0.020915385335683823, 0.03066762536764145]
2_9      8      violated      1.1357986420002817
```

Figure 6: nnenum unsafe output

The next phase involved testing all properties, aiming to evaluate the neural network's behavior across various configurations and constraints. The final verified accuracy stands at approximately 74.73%. Upon comparing the results from alpha-beta-CROWN and nnenum for the 186 instances tested, both tools identified 139 instances as safe, meaning that the property held true, and 47 instances as unsafe. This information is summarized in table 1.

Tool	Verified	Falsified	Penalty	Score
alpha-beta-CROWN	139	47	0	1860
nnenum	139	47	0	1860

Table 1: Benchmark 2023-ACAS-Xu

Regarding the verification times for alpha-beta-CROWN, the mean time taken for verifying all instances was approximately 2.58 seconds, with the maximum time being around 69.08 seconds. In terms of the employed methods, specifically the

Branch and Bound method (BAB), 4 instances were marked as unsafe. In contrast, utilizing the Projected Gradient Descent (PGD) method, 47 instances were identified as unsafe.

As for nenum, the performance achieved was quite similar, where all the instances have been verified, on average, in approximately 2.3 seconds with a maximum verification time of 61.58.

A more detailed overview is available on GitHub⁴, featuring two CSV files that present results for both alpha-beta-CROWN and nenum. These files detail the satisfiability and verification times for every tested combination. In addition, counter examples were generated and uploaded for the instances marked as unsafe.

5 Conclusions

This study presents the results of the evaluation of state of the art verification methods on the ACAS-Xu benchmark. The aim was to replicate the results from the VNN-COMP 2023[2] competition with the usage of alpha-beta-CROWN and nenum. The findings revealed that out of 186 instances examined, 47 were identified as unsafe. It's worth noting a significant difference in the results when running alpha-beta-CROWN compared to the competition, as there were no penalties. The instance that did result in a penalty was classified as unsafe. The verification process used the same parameters as those used in the competition, with the source files obtained directly from the official GitHub repository ⁵.

⁴<https://github.com/razvanmaciovan/Proiect-VF>

⁵https://github.com/ChristopherBrix/vnncomp2023_benchmarks/tree/main/benchmarks/acasxu

References

- [1] Stanley Bak. nenum: Verification of relu neural networks with optimized abstraction refinement. In *NASA Formal Methods: 13th International Symposium, NFM 2021, Virtual Event, May 24–28, 2021, Proceedings*, page 19–36, Berlin, Heidelberg, 2021. Springer-Verlag.
- [2] Christopher Brix, Stanley Bak, Changliu Liu, and Taylor T. Johnson. The fourth international verification of neural networks competition (vnn-comp 2023): Summary and results, 2023.
- [3] Guy Katz, Clark Barrett, David L Dill, Kyle Julian, and Mykel J Kochenderfer. Reluplex: An efficient SMT solver for verifying deep neural networks. In *Computer Aided Verification: 29th International Conference, CAV 2017, Heidelberg, Germany, July 24–28, 2017, Proceedings, Part I 30*, pages 97–117. Springer, 2017.
- [4] Guy Katz, Derek A. Huang, Duligur Ibeling, Kyle Julian, Christopher Lazarus, Rachel Lim, Parth Shah, Shantanu Thakoor, Haoze Wu, Aleksandar Zeljić, David L. Dill, Mykel J. Kochenderfer, and Clark Barrett. The marabou framework for verification and analysis of deep neural networks. In Isil Dillig and Serdar Tasiran, editors, *Computer Aided Verification*, pages 443–452, Cham, 2019. Springer International Publishing.
- [5] Mykel J Kochenderfer and JP Chryssanthacopoulos. Robust airborne collision avoidance through dynamic programming. *Massachusetts Institute of Technology, Lincoln Laboratory, Project Report ATC-371*, 130, 2011.
- [6] Zhouxing Shi, Qirui Jin, J Zico Kolter, Suman Jana, Cho-Jui Hsieh, and Huan Zhang. Formal verification for neural networks with general nonlinearities via branch-and-bound. *2nd Workshop on Formal Verification of Machine Learning (WFVML 2023)*, 2023.
- [7] Shiqi Wang, Huan Zhang, Kaidi Xu, Xue Lin, Suman Jana, Cho-Jui Hsieh, and J Zico Kolter. Beta-CROWN: Efficient bound propagation with per-neuron split constraints for complete and incomplete neural network verification. *Advances in Neural Information Processing Systems*, 34, 2021.
- [8] Kaidi Xu, Zhouxing Shi, Huan Zhang, Yihan Wang, Kai-Wei Chang, Minlie Huang, Bhavya Kailkhura, Xue Lin, and Cho-Jui Hsieh. Automatic perturbation analysis for scalable certified robustness and beyond. *Advances in Neural Information Processing Systems*, 33, 2020.
- [9] Kaidi Xu, Huan Zhang, Shiqi Wang, Yihan Wang, Suman Jana, Xue Lin, and Cho-Jui Hsieh. Fast and Complete: Enabling complete neural network verification with rapid and massively parallel incomplete verifiers. In *International Conference on Learning Representations*, 2021.
- [10] Huan Zhang, Shiqi Wang, Kaidi Xu, Linyi Li, Bo Li, Suman Jana, Cho-Jui Hsieh, and J Zico Kolter. General cutting planes for bound-propagation-based neural network verification. *Advances in Neural Information Processing Systems*, 2022.

- [11] Huan Zhang, Tsui-Wei Weng, Pin-Yu Chen, Cho-Jui Hsieh, and Luca Daniel. Efficient neural network robustness certification with general activation functions. *Advances in Neural Information Processing Systems*, 31:4939–4948, 2018.