

Formal Verification of Neural Networks

¹Thanapong Sommart, ²Borja Fernández Adiego

¹Sirindhorn International Institute of Technology, Pathum Thani, Thailand

²CERN, Geneva, Switzerland

Abstract

With the increasing popularity of neural networks, it is also important to make sure that at least some properties can be guaranteed for neural networks, especially if safety is a major concern for their applications. In this work, techniques and tools for formal verification specifically made for neural networks are studied. The tools are top performers in the VNN-COMP 2022, which is a competition specifically for formally verifying neural networks. By providing a neural network model in the ONNX format, and a specification in the VNNLIB format, the tools can find whether there exists a case where the specification is satisfied, given the model. With this result, several properties can be verified for different applications of neural networks.

Keywords

neural network, formal verification, specification

1 Introduction

Neural networks with various architectures and sizes, as shown in Figure 1, are becoming ubiquitous in many fields and applications. At CERN, several neural networks are being developed for control systems for LHC, such as cooling tower control systems [1] and BLM sensor instance segmentation. However, reliability and safety of neural networks are heavily concerned due to the “black box” nature of their behavior. Therefore, in order to guarantee or verify that a certain scenario will never happen, formal verification for neural networks is required, especially in critical systems.

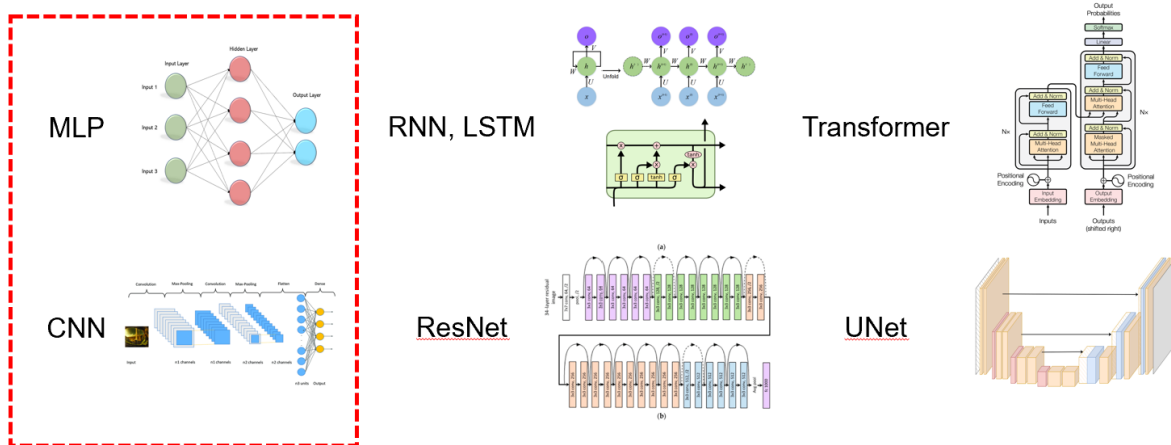


Figure 1: Example of NN architectures.

Among the variety of neural networks, a multi-layer perceptron (MLP) and a convolutional neural network (CNN) are the most popular architectures of neural networks for most control systems out there, so most of the experiments will mainly focus on these two architectures.

2 State of the Art

In order to discover available techniques and tools for formal verification, the VNN-COMP 2022 [2] is chosen to be the main entry point of this study. The VNN-COMP stands for the Verification of Neural Networks Competition, which has been held annually for three years, and has had many active researchers in this field as participants.

From the most recent VNN-COMP, we have selected three tools which achieved the best performances, participated in most (if not all) benchmarks, and provided sufficient documentation for our application, which includes:

1. **alpha-beta CROWN** [4–7], which utilizes linear bound propagation and the branch and bound [8] method for adversarial search
2. **nnenum** [9], which uses zonotopes for over-approximation and geometric path enumeration [10] for efficiently splitting ReLU networks
3. **VeriNet** [11–13], which implements symbolic interval propagation and a branch-and-bound-based method.

2.1 Background

Although each tool is implemented independently, the common approach to verify a neural network is to efficiently compute the output bounds of neural network outputs by relaxing activation functions and non-linear operations. This concept can be illustrated in Figure 2. Note that the actual rule for relaxation varies among the tools and research. Most works such as [3, 14–16] focus on only ReLU activation, while others such as [17] try to generalize the relaxation for more activations like sigmoid, tanh, etc. After that, the verification can be formulated as a linear programming (LP) problem, which can be solved using third-party LP-solving tools.

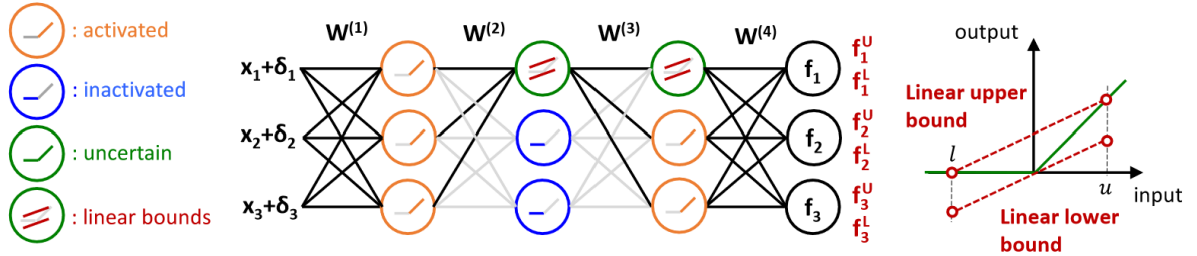


Figure 2: Example of linear relaxation for ReLU networks proposed by [3].

2.2 Standard Format

```
(declare-const X_0 Real)      (assert (>= X_0 20.000000))      (assert (or
(declare-const X_1 Real)      (assert (<= X_0 25.000000))      (and (<= Y_2 Y_0))
(declare-const X_2 Real)      (assert (>= X_1 23.000000))      (and (<= Y_2 Y_1))
(declare-const Y_0 Real)      (assert (<= X_1 27.000000))      ))
(declare-const Y_1 Real)      (assert (>= X_2 8.000000))
(declare-const Y_2 Real)      (assert (<= X_2 21.000000))
```

As the standard for the VNN-COMP 2022, the tools are compatible with neural network models in the Open Neural Network Exchange (ONNX) format, which can be easily converted from other file

formats from other machine learning frameworks. For specifications, the VNNLIB format, specifically created for the competition, is used. This specification format consists of variable declarations and assertion statements, as shown above.

2.3 Verification Result

By providing an ONNX model file and a VNNLIB specification file to each tool, the result indicates whether a counter-example, a case where all conditions in the specification is true given the model, exists. If a counter-example exists, the result is denoted SAT. Otherwise, it is denoted UNSAT. In some situations, however, a tool may not be able to conclude the result for the verification, and will provide UNKNOWN as a result. In addition, since the tools are designed for the competition where scores also depend on runtime, if a tool is unable to finish verification within a specified period of time, the algorithm is terminated and TIMEOUT will be provided as the result.

In practice, these verification results can be interpreted as guarantee for certain properties, depending on what the specifications mean for the users. Some of these interpretations will be demonstrated in Section 3. The method of formulating problems for verification will be a bit different, depending on the quantifier of the property we wish to verify for a neural network f , where $y = f(x)$.

2.3.1 Existential Verification

If we aim to guarantee that an output condition $v(y)$ may hold given closed bounds of possible inputs $u(x)$, meaning our property P is in the form

$$P \equiv \exists x, u(x) \wedge v(y)$$

then we can directly formulate the specification as $H \equiv P$. If the result is SAT, then the property P is guaranteed for f .

2.3.2 Universal Verification

If we aim to guarantee that an output condition $v(y)$ always holds given closed bounds of possible inputs $u(x)$, meaning our property P is in the form

$$P \equiv \forall x, u(x) \rightarrow v(y)$$

then, since the verification tools can only find counter-examples, we instead try to guarantee a possibility of any violations of P , so the specification becomes

$$\begin{aligned} H &\equiv \sim P \\ &\equiv \sim (\forall x, u(x) \rightarrow v(y)) \\ &\equiv \exists x, \sim (\sim u(x) \vee v(y)) \\ &\equiv \exists x, u(x) \wedge \sim v(y) \end{aligned}$$

If the result is UNSAT, then the property P is guaranteed for f .

2.4 GUI Application

To conduct a study on neural network verification using the available tools, a GUI is developed to provide a unified pipeline, as shown in Figure 3, for using all of the tools to verify a given model and a given specification. The GUI provides a convenient way to choose ONNX and VNNLIB files for verification with all (or each) of the tools under a specified timeout. It also shows runtime used by each verification tool, and a counter-example provided by each tool providing SAT as a verification result.



Figure 3: Overall pipeline of neural network verification

3 Experiments

To demonstrate a range of properties neural network formal verification can guarantee, we mainly consider the neural network from [1]. Although we have successfully perform verification for other models such as a 2D-CNN MNIST classifier and a 1D-BLM signal classifier, their properties are encoded in similar fashion to our example in this section.

3.1 Overview

The example model, as shown in Figure 4(a), consists of fully-connected layers mapping three temperature inputs from sensors within the LHC cooling tower to two separate outputs: a fan speed and the probabilities for the three possible modes of operation. Therefore, this model can be view or split into two MLPs where the weights in the first two layers are the same. The model uses ReLU activation at every layer except at the last classification layer, where softmax activation is used. For this experiment, since we would like to utilize all three tools, the softmax activation is removed, making the classification output become logits scores instead. This practically makes no difference, since we can still determine the classification result using the highest logits score.

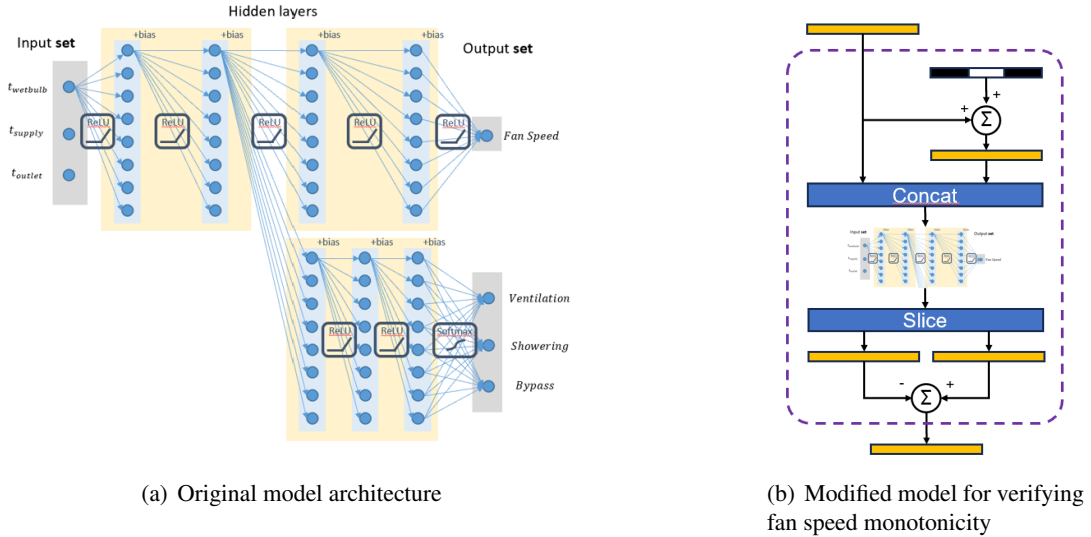


Figure 4: Overview of LHC cooling tower system neural network for formal verification

3.2 Verifiable Properties

3.2.1 Reachability

This property is about whether there exists a set of input temperatures within the operating region which can produce the target output, such as the following:

- The second mode of operation can be activated during operation.
(the second logits score can be the highest score)
- The fan speed will never reached 20%-40% during operation.

3.2.2 Robustness

This property (may also be called *monotonicity* in the context of classification) is about whether all outputs will follow the expectation for all input temperatures within the operating region which can produce an output within a target interval, such as the following:

- The LHC cooling tower will always be in the first mode of operation during a certain state.
(the first logits score will always be the highest score)
- The fan speed will remain within 60%-80% during operation.

3.2.3 Monotonicity

This property (in the context of regression problems) is about whether an output at a specific dimension will always strictly increase or decrease if an input at another specific dimension increases. However, to make this work, the model needs to be encapsulated with additional custom operations as a new model for verification. This is illustrated in Figure 4(b).

4 Discussion

4.1 Performance Comparison

After putting the available neural network verification tools in Section 2 to use, it can be observed that for different neural network models and different specifications, the performance can vary among the tools, currently with no obvious patterns. However, the overall performance for each tool can be summarized as shown in Table 1 shown below.

Table 1: Overall advantages and disadvantages of experimented tools

Verification Tool	Advantages	Disadvantages
alpha-beta CROWN	Has low runtime	Needs complicated configurations
nnenum	Works with custom built networks	Provides some unknown results
	Is fast and easy to use	Only works with ReLU activation
VeriNet	Barely provides unknown results	
	Works with most activations	Does not support certain operations
	Can utilize multiprocessing	Has higher runtime

4.2 Limitations

Despite the successful attempts at using these verifiers for neural network verification in Section 3, there are still limitations in using these tools for neural network verification:

4.2.1 *Specification Expressiveness*

Since the tools are designed for verifying output properties given input properties, they are only capable of working with a single pass through a neural network. they currently do not support verifying properties involving temporal logics (i.e. properties involving loops). In addition, since the parsers are designed for properties given by the VNN-COMP, which are mostly not too complicated, the tools may not work with too many nested conjunctions or disjunctions in the VNNLIB format.

4.2.2 *Architecture Support*

Neural network verification requires verification tools to both make a linear approximation of each operation in a neural network and effectively "split" the model to make verification feasible, some complex neural network architectures, such as RNNs, LSTMs, and transformers [18], are not supported.

Fortunately, neural network formal verification for more complex architectures, especially recurrent nodes, are being studied and actively researched on. For example, [19] proposed a method for verifying LSTM models with non-linear activations. In fact, alpha-beta CROWN, one of the three tools from Section 2, claims that it can also work with recurrent connections, but this currently has not been tested in this work.

4.2.3 *Model Adjustments*

In case the verifiers indicate that a model does not have a desired property, they cannot fully explain the cause of such result, and cannot provide us how to correct or adjust the model so that the property can be guaranteed. However, an interesting approach is presented by [15], which demonstrated an increase in robustness of a MLP model when trained with a custom robust loss, essentially letting the model learn to be robust to noise since during training.

5 Conclusion

In this project, several top-performing tools from the VNN-COMP 2022 for neural network verification are utilized to study a range of properties which can be verified for several applications of neural networks. Further development includes applying the verification tools to other neural networks deployed at CERN, discovering more techniques to express safety properties as specifications, and exploring verification techniques for more complex network structures such as RNNs, transformers, etc.

Acknowledgements

We wish to thank Xavier Eugen Fink for providing knowledge and advice relevant to the project, especially theoretical concepts of formal verification.

Bibliography

- [1] Lopez-Miguel, Ignacio D., Borja Fernández Adiego, Faiq Ghawash, and Enrique Blanco Viñuela. "Verification of Neural Networks Meets PLC Code: An LHC Cooling Tower Control System at CERN." In International Conference on Engineering Applications of Neural Networks, pp. 420-432. Cham: Springer Nature Switzerland, 2023.
https://link.springer.com/chapter/10.1007/978-3-031-34204-2_35
- [2] Müller, Mark Niklas, Christopher Brix, Stanley Bak, Changliu Liu, and Taylor T. Johnson. "The third international verification of neural networks competition (VNN-COMP 2022): summary and results." arXiv preprint arXiv:2212.10376 (2022).
<https://arxiv.org/abs/2212.10376>.

- [3] Weng, Lily, Huan Zhang, Hongge Chen, Zhao Song, Cho-Jui Hsieh, Luca Daniel, Duane Boning, and Inderjit Dhillon. "Towards fast computation of certified robustness for relu networks." In International Conference on Machine Learning, pp. 5276-5285. PMLR, 2018.
<https://dl.acm.org/doi/10.1145/3290354>.
- [4] Xu, Kaidi, 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): 1129-1141.
<https://proceedings.neurips.cc/paper/2020/hash/0cbc5671ae26f67871cb914d81ef8fc1-Abstract.html>
- [5] Zhang, Huan, 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 (2018).
<https://proceedings.neurips.cc/paper/2018/hash/d04863f100d59b3eb688a11f95b0ae60-Abstract.html>
- [6] Wang, Shiqi, 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 neural network robustness verification." *Advances in Neural Information Processing Systems* 34 (2021): 29909-29921.
https://proceedings.neurips.cc/paper_files/paper/2021/hash/fac7fead96dafceaf80c1daffeae82a4-Abstract.html
- [7] Zhang, Huan, 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* 35 (2022): 1656-1670.
https://proceedings.neurips.cc/paper_files/paper/2022/hash/0b06c8673ebb453e5e468f7743d8f54e-Abstract-Conference.html
- [8] Bunel, Rudy R., Ilker Turkaslan, Philip Torr, Pushmeet Kohli, and Pawan K. Mudigonda. "A unified view of piecewise linear neural network verification." *Advances in Neural Information Processing Systems* 31 (2018).
https://proceedings.neurips.cc/paper_files/paper/2018/hash/be53d253d6bc3258a8160556dda3e9b2-Abstract.html
- [9] Bak, Stanley. "nenum: Verification of relu neural networks with optimized abstraction refinement." In *NASA Formal Methods Symposium*, pp. 19-36. Cham: Springer International Publishing, 2021.
https://link.springer.com/10.1007/978-3-030-76384-8_2
- [10] Bak, Stanley, Hoang-Dung Tran, Kerianne Hobbs, and Taylor T. Johnson. "Improved geometric path enumeration for verifying relu neural networks." In *Computer Aided Verification: 32nd International Conference, CAV 2020, Los Angeles, CA, USA, July 21–24, 2020, Proceedings, Part I* 32, pp. 66-96. Springer International Publishing, 2020.
http://link.springer.com/10.1007/978-3-030-53288-8_4
- [11] Henriksen, Patrick, and Alessio Lomuscio. "DEEPSPLIT: An Efficient Splitting Method for Neural Network Verification via Indirect Effect Analysis." In *IJCAI*, pp. 2549-2555. 2021.
<https://www.ijcai.org/proceedings/2021/351>
- [12] Henriksen, Patrick, and Alessio Lomuscio. "Efficient neural network verification via adaptive refinement and adversarial search." In *ECAI 2020*, pp. 2513-2520. IOS Press, 2020.
<https://ebooks.iospress.nl/doi/10.3233/FAIA200385>
- [13] Henriksen, Patrick, Kerstin Hammernik, Daniel Rueckert, and Alessio Lomuscio. "Bias field robustness verification of large neural image classifiers." In *Proceedings of the 32nd British Machine Vision Conference (BMVC21)*. BMVA Press, vol. 1, p. 2. 2021.

- <https://www.bmvc2021-virtualconference.com/assets/papers/1291.pdf>
- [14] Katz, Guy, 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, pp. 97-117. Springer International Publishing, 2017.
https://link.springer.com/chapter/10.1007/978-3-319-63387-9_5
 - [15] Wong, Eric, and Zico Kolter. "Provable defenses against adversarial examples via the convex outer adversarial polytope." In International conference on machine learning, pp. 5286-5295. PMLR, 2018.
<http://proceedings.mlr.press/v80/wong18a.html?ref=https://githubhelp.com>
 - [16] Singh, Gagandeep, Timon Gehr, Markus Püschel, and Martin Vechev. "An abstract domain for certifying neural networks." Proceedings of the ACM on Programming Languages 3, no. POPL (2019): 1-30.
<https://dl.acm.org/doi/abs/10.1145/3290354>
 - [17] Singh, Gagandeep, Timon Gehr, Matthew Mirman, Markus Püschel, and Martin Vechev. "Fast and effective robustness certification." Advances in neural information processing systems 31 (2018).
https://proceedings.neurips.cc/paper_files/paper/2018/hash/f2f446980d8e971ef3da97af089481c3-Abstract.html
 - [18] Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. "Attention is all you need." Advances in neural information processing systems 30 (2017).
https://proceedings.neurips.cc/paper_files/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html
 - [19] Moradkhani, Farzaneh, Connor Fibich, and Martin Fränzle. "Verification of LSTM Neural Networks with Non-linear Activation Functions." In NASA Formal Methods Symposium, pp. 1-15. Cham: Springer Nature Switzerland, 2023. https://link.springer.com/chapter/10.1007/978-3-031-33170-1_1