Formal Verification for Safer Learning-Enabled Autonomous Systems

R. Desmartin, Edinburgh Centre for Robotics - Heriot-Watt University

Supervised by E. Komendskaya, G. Passmore, K. Stark

Deep Neural Networks for Robotics and Autonomous Systems

Deep neural networks (DNNs) have achieved outstanding results in computationally intensive tasks, with breakthrough applications in many areas of robotics and autonomous systems, e.g. perception, planning and decision making, and control. Thanks to their performance they have been deployed in safety-critical domains such as autonomous vehicles or medicine [Grigorescu et al., 2020].

Figure 1. Boeing Autonomy Testbed Aircraft, Cessna Caravan 208B, used for deploying a learning-enabled controller [Cofer et al., 2022]

For instance, **ACAS Xu**, an unmanned aircraft collision avoidance system, was compressed without loss of performance by replacing look-up tables with a set of DNNs [Julian et al., 2016]. Cofer et al. [2022] deployed and tested the compressed ACAS Xu in an autonomous aircraft in various potential collision scenarios (see Figure 1).

Formal Verification of Deep Neural Networks

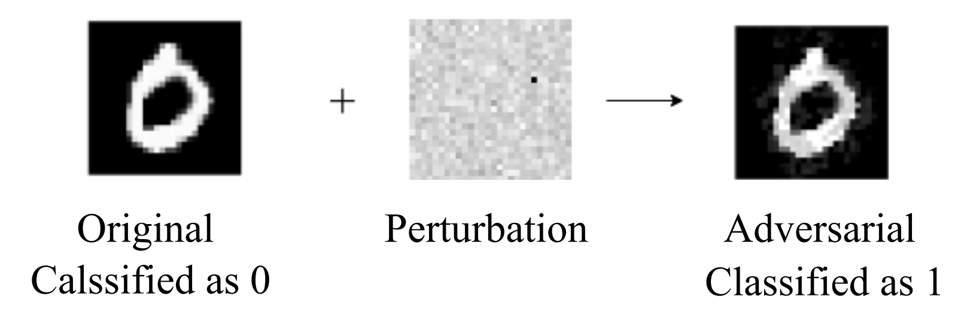


Figure 2. Adversarial perturbations are indistinguishable by humans but can fool a DNN classifier.

Despite their high performance, DNNs are known to be sensitive to small changes in their inputs, whether from noisy inputs or malicious perturbations [Szegedy et al., 2013]. As a result, in recent years the verification community has developed **formal verification tools for DNNs**, i.e. tools that allow reasoning on DNNs' **critical properties** using **explicit mathematical modelling** and **sound logic**.

The most commonly studied property is **robustness**, defined as the ability of a network, usually a classifier, to give stable outputs for all L_p-norm bounded perturbations around a correctly classified input.

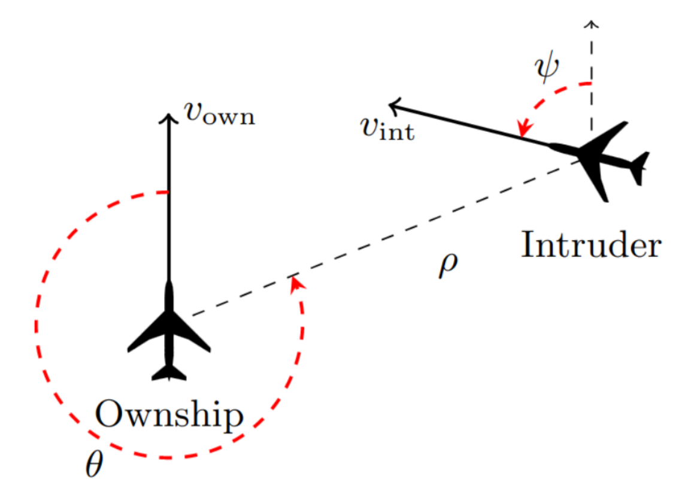


Figure 3. Geometry of the ACAS Xu controller [Katz et al., 2017]

Katz et al. [2017] formulated and verified that the compressed ACAS Xu controller satisfied some domain-specific safety properties, e.g. *"If an intruder is directly ahead and is moving towards the ownship, the controller will advise to turn left or right."*

Industrial Applications of Formal Verification

Formal verification has known successful applications in industries where errors can have a significant impact, such as hardware manufacturing, cryptography or transportation [Hunt et al., 2017].

Imandra is an industrial verification tool that has been successfully applied in finance, for instance by formalising the FIX banking exchange protocol [Passmore, 2021]. Its logic is based on a pure subset of OCaml, and its interactive proof mode is based on a typed, higher-order lifting of the Boyer-Moore waterfall for inductive reasoning [Boyer and Moore].

Once the model is verified, Imandra has the capacity to export it to native OCaml while preserving its semantics. The exported program can then benefit from native OCaml compilation optimisations.

Several uses of Imandra for guaranteeing safety properties in autonomous systems have been explored.

Imandra-ROS Interface

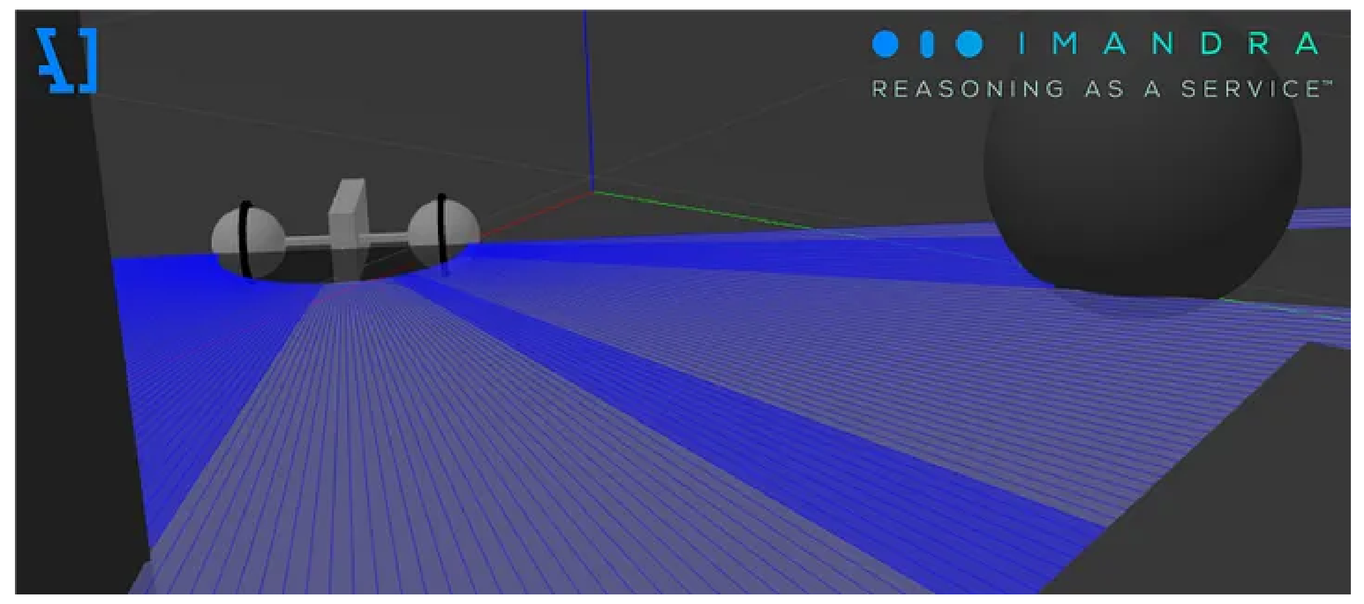
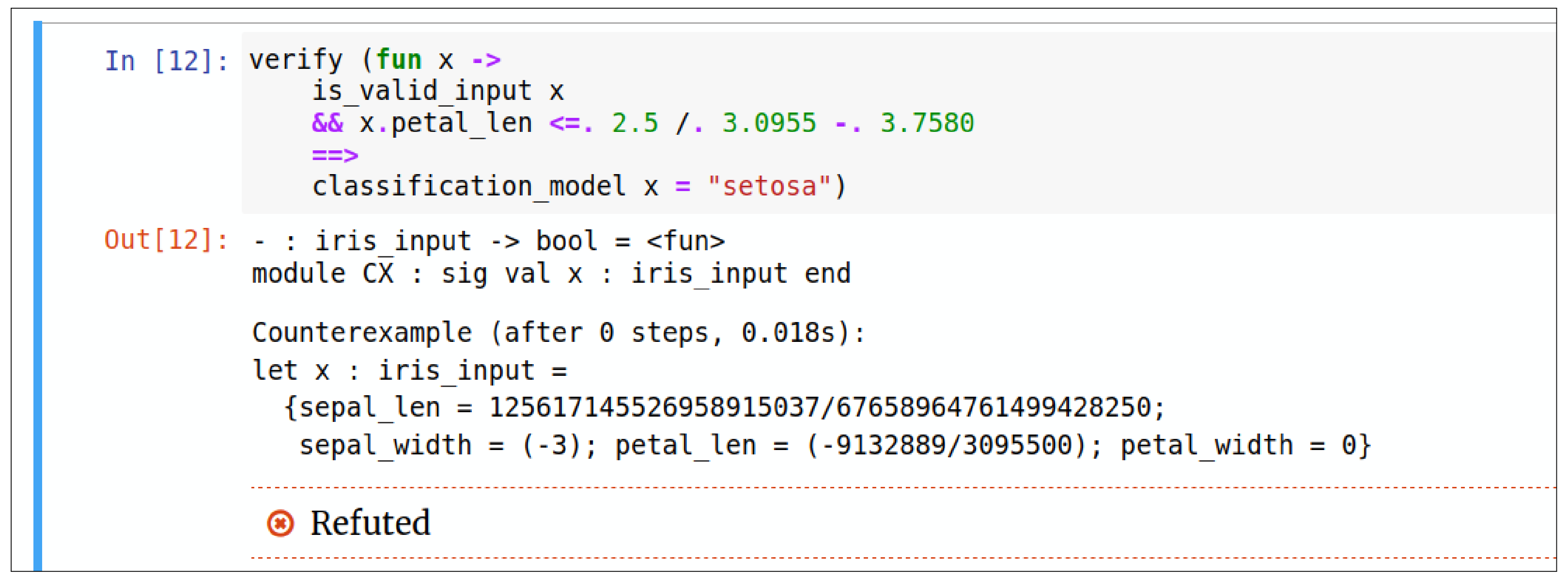


Figure 4. Simulation of a robot whose controller is implemented and verified in Imandra [Kanishev, 2018]

Kanishev [2018] presents an Imandra-ROS interface, that allows to implement a ROS node to control an autonomous robot in Imandra. The controller is implemented using the Imandra Modelling Language (IML), Imandra's standardised way of modelling message-driven systems. The model can be formally verified against a safety specification in Imandra's reasoning environment, and then compiled to a native OCaml program. Thanks to the Imandra-ROS interface, the controller can then be deployed to control a ROS-operated robot, receiving sensor data and sending instructions in real time.

DNN Verification in Imandra



```
In [12]: verify (fun x ->
             is_valid_input x
             && x.petal_len <=. 2.5 /. 3.0955 -. 3.7580
             ==>
             classification_model x = "setosa")

Out[12]: - : iris_input -> bool = <fun>
         module CX : sig val x : iris_input end

         Counterexample (after 0 steps, 0.018s):
         let x : iris_input =
           {sepal_len = 125617145526958915037/67658964761499428250;
            sepal_width = (-3); petal_len = (-9132889/3095500); petal_width = 0}

         ⊗ Refuted
```

Figure 5. Example of a verification query's execution in Imandra. Note that Imandra proved that the property doesn't hold, and provided an executable counter-example as a result.

We formalised common DNN architectures (multi-layer perceptron and convolutional) in Imandra and used its SMT solving and induction reasoning capabilities to verify that they satisfy some logical properties [Desmartin et al., 2022]. Imandra's logic allowed to state and verify high-level meta-properties about the DNNs, i.e. properties that apply to all networks with the same structure, which are unsupported by state-of-the-art verifiers. On the other hand, scaling remained a challenge.

Current Work: A Verified Proof Checker for DNN Verification

DNN verifiers are themselves complex pieces of software and are susceptible to implementation errors or floating-point imprecision. DNN verifiers usually verify logical properties that negates a safety specification. If a satisfying assignment is found, it means that the safety spec doesn't hold, if no counter-example is found, it means that the safety property holds. As a result, it is easy to check if a satisfying assignment is correct, by running it through the DNN, but it is not trivial to check that no satisfying assignment exist.

Proof production is a well-known technique to guarantee the result of verification tools, used especially in SMT solvers. The verifier produces a proof of its result, which can be checked externally by a trusted **proof checker** to guarantee its result's correctness. The proof checking procedure is usually simpler than the verification one. Such proof production has been implemented for the Marabou DNN verifier, but the existing proof checker doesn't offer formal guarantee of correctness [Isac et al., 2022].

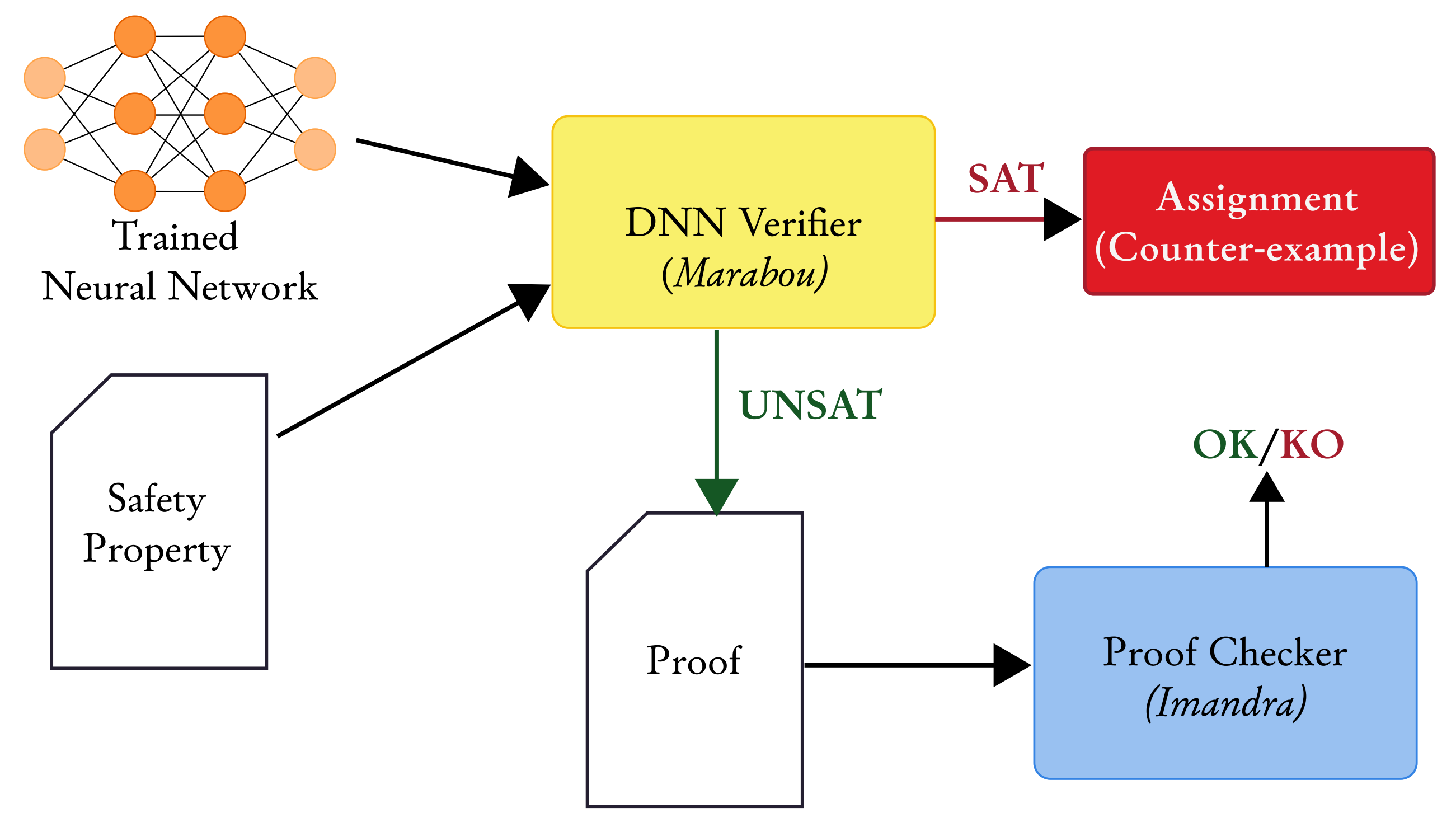


Figure 6. Proof-checking workflow

Our current work focuses in **implementing** a proof checker in Imandra, **specifying its soundness** property and **verifying** it.

References

R. S. Boyer and J. S. Moore. The Boyer-Moore Theorem Prover. https://www.cs.utexas.edu/users/moore/best-ideas/nqthm/index.html.

D. Cofer, R. Sattigeri, I. Amundson, J. Babar, S. Hasan, E. W. Smith, K. Nukala, D. Osipychev, M. A. Moser, J. L. Paunicka, D. D. Margineantu, L. Timmerman, and J. Q. Stringfield. Flight Test of a Collision Avoidance Neural Network with Run-Time Assurance. In *2022 IEEE/AIAA 41st Digital Avionics Systems Conference (DASC)*, pages 1–10, Sept. 2022.

R. Desmartin, G. O. Passmore, E. Komendantskaya, and M. Daggit. CheckINN: Wide Range Neural Network Verification in Imandra. In *PPDP 2022: 24th International Symposium on Principles and Practice of Declarative Programming, Tbilisi, Georgia, September 20 - 22, 2022*, pages 3:1–3:14. ACM, 2022.

S. Grigorescu, B. Trasnea, T. Cocias, and G. Macesanu. A survey of deep learning techniques for autonomous driving. *Journal of Field Robotics*, 37(3):362–386, 2020. ISSN 1556-4967.

W. A. Hunt, M. Kaufmann, J. S. Moore, and A. Slobodova. Industrial hardware and software verification with ACL2. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 375(2104):20150399, Sept. 2017.

O. Isac, C. W. Barrett, M. Zhang, and G. Katz. Neural network verification with proof production. *2022 Formal Methods in Computer-Aided Design (FMCAD)*, pages 38–48, 2022.

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

K. Kanishev. Imandra interface to Robot OS: Part I. https://medium.com/imandra/imandra-interface-to-robot-os-part-i-9f3888c5c3a1, Aug. 2018.

G. Katz, C. W. Barrett, D. L. Dill, K. Julian, and M. J. Kochenderfer. Reluplex: An Efficient SMT Solver for Verifying Deep Neural Networks. In R. Majumdar and V. Kuncak, editors, *Computer Aided Verification - 29th International Conference, CAV 2017, Heidelberg, Germany, July 24-28, 2017, Proceedings, Part I*, volume 10426 of *Lecture Notes in Computer Science*, pages 97–117. Springer, 2017.

G. O. Passmore. Some Lessons Learned in the Industrialization of Formal Methods for Financial Algorithms. In M. Huisman, C. Păsăreanu, and N. Zhan, editors, *Formal Methods*, Lecture Notes in Computer Science, pages 717–721, Cham, 2021. Springer International Publishing. ISBN 978-3-030-90870-6.

C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. J. Goodfellow, and R. Fergus. Intriguing properties of neural networks. *CoRR*, Dec. 2013.

rhd2000@hw.ac.uk