Assuring Safety-Critical Machine Learning Enabled Systems: Challenges and Promise

Alwyn E. Goodloe
NASA Langley Research Center
Hampton, Va. USA
a.goodloe@nasa.gov

***Abstract*—Machine learning is increasingly being used in safety-critical systems, where the public safety requires a rigorous assurance process. We shall outline how assurance processes work for conventional systems and identify the primary difficulty in applying them to machine learning enabled systems. We will then outline a path forward including identifying where considerable basic research remains.**

***Index Terms*—safety-critical systems, assurance, machine-learning, certification.**

I. Introduction

Safety-critical systems, such as aircraft, automobiles, and medical devices are systems whose failure could result in loss of life, significant property damage, or damage to the environment [1], [2]. The grave consequences of failure have compelled industry and regulatory authorities to adopt conservative design approaches and exhaustive verification and validation (V&V) procedures to prevent mishaps. In addition, strict licensing requirements are often placed on human operators of many safety-critical systems. In practice, the verification and validation of avionics and other safety-critical software systems relies heavily on traceability to requirements and system predictability. Currently, in civil aviation, nuclear power, and similar highly regulated areas, there is no regulatory guidance for assuring artificial intelligence (AI) and similar approaches that do not exhibit predictable behavior at certification.

Technological advances such as the significant progress in machine learning are enabling the development of increasingly autonomous (IA) cyber-physical systems (CPS) that modify their behavior in response to the external environment and learn from their experience [3]. Machine learning (ML) is being employed to enable autonomous systems that operate in the real world, but many implementations lack the salient features of traceability and predictability. Moreover, these systems pose new dangers to public safety, especially when they encounter unexpected objects or events [4]–[8]. Hence, it will be necessary to assure system safety at the same level as existing systems if they are not to impose an unacceptable danger to public safety [9]. Efforts are underway to write standards and guidelines to govern the use of this technology such as the ANSI/UL 4600 Standard for Safety for the Evaluation of Autonomous Products [10] and SAE G 34 Artificial Intelligence in Aviation, but significant technical barriers must be overcome.

In this paper, we shall provide a brief overview of the processes and practices for assuring conventional software enabled systems focusing on the domain of civil aviation, which has an exemplary safety record. We shall discuss the challenges of assuring machine learning enabled systems and discuss some of the tools and techniques that are being proposed for this task and their drawbacks. Although it may seem an impossible task to assure machine learning enabled safety-critical systems within current assurance frameworks, we will discuss how particular classes of problems are within reach while others are likely to remain basic research challenges for the foreseeable future.

II. Assuring Conventional Systems

Years of experience at building safety-critical software systems such as aircraft and nuclear power systems have yielded analysis, design, and development practices that have produced extraordinarily safe systems such as the current air transportation system. So much so that the public demands that technological advances not lower the level of safety they have come to expect. In this section, we will give a high-level overview of design and development processes used for civil transport aircraft.

Like all software systems development, engineering safety-critical systems begins with ascertaining require-

ments, but in addition, there is a need for one or more safety analyses such as:

- A process used to assess risk such as a hazard analysis for the identification of different types of hazards.
- A process to identify potential failure modes in a system and their causes and effects.

In civil aviation, safety analyses are carried out whenever there are changes made to the system. The functional requirements and safety analyses together flow into the system specification, system architecture, and design. A functional specification precisely states what the system is to do, usually in terms of a formal relation of system output given a specified input. Hence it is also possible to deduce what constitutes undesired behavior. We often define functional correctness as follows, given a specification of initial conditions ψ and a specification of system requirements ϕ and a system S, if ψ is true and we execute S, then S will terminate in a state where ϕ is true. If it is possible to write a functional specification ϕ precisely stating what the system is to do, it is also possible to deduce what constitutes undesired behavior. When we can refine requirements into such a specification we call them *actionable specifications*.

The safety analysis will determine what faults the system will be expected to sustain and still operate safely. This is called the fault model of the system. Faults and failures are often mitigated at the architectural level by employing sufficient redundancy. Engineers have to demonstrate traceability back to the original requirements and safety analyses at each refinement step of the development process. These practices are often codified in guidelines [11], [12]. Software implementations tend to adhere to very conservative guidelines [13], [14] that constrain non-determinism and ensure bounds on resource consumption by disallowing dynamic memory allocation and recursion. Although these restrictions may constrain the design space, they make the task of assuring such systems tractable.

The assurance processes for safety-critical systems typically uses testing to demonstrate that the system meets the requirements, that there is no unintended behavior, and that the system tolerates specified faults. Coverage metrics are used that measure how well test inputs exercise the code. In particular, they aim to show the degree to which the test inputs execute all branches of the code. In civil aviation, the DO-178C [13] guidelines require that the most critical software to undergo modified condition/decision coverage (MC/DC) testing [15], where

- Each entry and exit point is invoked.
- Each decision takes every possible outcome.
- Each condition in a decision takes every possible outcome.

In addition to testing, formal methods based tools are increasingly being used for certification credit [16].

III. A 10,000 foot View of Machine Learning

When building conventional systems, one refines requirements into an actionable specification that is then refined into program logic and implemented in a programming language. The resulting program consumes data as input as it executes the program logic, which often makes decisions based on that input. In contrast, machine learning systems are constructed by providing the system with examples that one can construe as a model of what is to be built. In the case of supervised learning, before the training begins the engineer selects a set of hyperparameter values that control the learning process. During training, model parameters are initialized to some base value. These parameters get updated based on the data sets drawn from examples and an optimization algorithm. In short, the model is defined by the model parameters that get updated as the system learns and the hyperparameters influence this process. The optimization process is intended to ensure that the system "generalizes" well, that is, providing the right output to input that was not given in the training data set. These models are typically implemented in neural networks, comprised of many layers of nodes with each node acting like an linear regression model computing outputs based on weights and a basis. Many neural networks today are comprised of thousands of nodes arranged in more than a hundred layers making them quite opaque. While supervised learning is probably the most common machine learning approach being proposed for safety-critical systems, reinforcement learning is very popular for solving planning problems. Reinforcement learning performs learning during operational deployment based on maximizing a reward function while interacting with the environment and thus learning from experience. The distinguishing feature of machine learning is that the data is the algorithm and, as we will see, this is what makes assurance so difficult.

Machine learning is often advertised as the approach to use when you do not know how to specify the system you want to build. Industrial use of machine learning spans almost every domain from advertising to finance

to agriculture. The more cautious safety-critical systems domain has been slower to adapt this technology especially those areas subject to strict regulatory oversight. As we have seen in Section II these systems are typically built using a requirements driven methodology that is difficult to apply when requirements that can be refined into actionable specifications are lacking. Yet the desire to build autonomous systems that need functionality such as perception, for which we currently do not know how to write actionable specification, has driven engineers to use machine learning as it is currently the option with the best performance. The problem in adapting the conventional assurance approaches is that other than large data sets of examples what constitutes a specification? Consider a machine learning based classifier for pictures of birds. So you may have many gigabytes of examples, but what exactly would constitute a specification of a hummingbird or a cardinal?

On the other hand, there are use cases where machine learning is used to replace conventionally built applications because it exhibits superior performance characteristics. For instance, a machine learning based aircraft fuel management systems might use significantly fewer computational resources than conventional solutions.

In addition to functional correctness, we often speak of the software safety properties defined as "something bad does not happen". Typical traditional safety properties are floating point arithmetic overflows and buffer overflows. Machine learning has its own basket of safety properties. For instance, neural network based perception classifiers have shown themselves to be sensitive to small changes in an image that may not even be recognizable to humans [17]. This phenomenon is called adversarial attack and systems that exhibit *adversarial robustness* do not exhibit such sensitive behavior. Moreover, there is strong evidence that neural network based classifiers can either be robust against adversarial attacks or accurate, but not both [18]. One of the more popular formulations of adversarial robustness [19] follows. Suppose the neural network N is a classifier associated with a given set of labels L, a given input x is classified as label $l \in L$ denoted $N(x)$. A neural network is said to be δ locally robust at input x_0 if small perturbations do not change the classification:

$$\forall x. ||x - x_0|| \leq \delta \Rightarrow N(x) = N(x_0).$$

Such safety properties are actionable specifications and are attractive because they are amenable to detection by automated tools. One of the reasons adversarial robustness has attracted so much attention is that it is one of the few such properties for which we have actionable specifications.

IV. Assurance Approaches

A number of approaches have been proposed for assurance of machine learning enabled systems. We will briefly survey five of these and assess their strengths and weaknesses.

A. Testing

Testing is the most well established approach used in assuring systems and it definitely plays a role in assuring machine learning enabled systems, but there are challenges. Given that the specification for a machine learning system is captured in the example data sets, it is very difficult to create test oracles [20]. Typically, a set of examples is held back from the training set and later used to test the performance of the machine learning system, but how the system responds to inputs it has not seen before is not precisely defined so at best there is some statistical argument to be made. There are numerous efforts to apply techniques that have been successful at testing conventional software [21], [22], but their efficacy is still being evaluated.

It is difficult to see how coverage metrics used in conventional software is easily transferred to this setting. A common coverage criterion for neural networks is that the test inputs are selected to ensure that all neurons are activated. The branching in neural network implementations is not very sophisticated so achieving such coverage is not difficult, but not very meaningful either.

Lacking neither actionable specifications nor good coverage metrics, it is not possible to test that a system performs its intended function and that there is no unintended behavior. Thus it is not possible to perform the kind of requirements based test driven assurance described in Section II. Discovering a new testing approach that provides the same level of confidence in assuring the system remains the subject of research.

B. Formal Methods

The application of techniques from the formal methods community to the verification of machine learning enabled systems is an active area of research. Consider a neural network that implements a function $y = f(x)$ for a bounded input domain $\mathcal{D}$. Given a correctness property $\phi(x, y)$ the goal is to show

$$\forall x \in \mathcal{D}. y = f(x) \Rightarrow \phi(x, y).$$

Rather than a direct proof, the problem is reformulated as a constraint problem [23] [24] . One approach is to recast the problem as a mixed integer programming problem [25]. An alternate approach is to recast the problem to be resolved by a Satisfiability Modulo Theories (SMT) solver [26]. RelUPlex [27] and its successor Marabou [28] are examples of this approach.

Abstract interpretation [29] is a static analysis technique that computes a sound and conservative over-approximation of a program by relating the concrete states of program to a more tractable abstract set of states and then automatically proving that the abstract program satisfies a given safety property. Researchers at ETH Zurich have recently been investigating how abstract interpretation can be applied to verifying neural networks [30], [31]. In this work, neural networks are represented as affine transformations guarded by logical constraints. Abstract interpretation tools have been applied to verify adversarial robustness.

These techniques work very well on small examples, but getting any of these approaches to scale remains a problem. As with testing, the biggest challenge is the need for actionable specifications. One of the reasons why so many research efforts focus on verifying the same properties is that there are simply so few known actionable specifications.

C. Runtime Verification

Runtime verification (RV) [32]–[34] is a verification technique that has the potential to enable the safe operation of safety-critical systems that are too complex to formally verify or fully test. In RV, the system is monitored *during execution*, to detect and respond to property violations that take place during the actual mission. The Copilot runtime verification framework [35], [36], developed by the author and colleagues at NASA, targets the runtime verification of safety-critical systems with a strong emphasis on certification [37]. Due to the probabilistic behavior of machine learning, runtime verification can help ensure that input that has never been seen does not result in unsafe behavior. Within NASA we have found that generating monitors from structured English requirements [38] makes runtime verification easier for working engineers.

Just as with testing and formal methods, there must be actionable specifications to check. For instance, a machine learning enabled autopilot may have a safety property saying it must stay within a well-defined geofence and this can be checked at runtime, but it is not possible to check that a classifier has properly detected a Persian cat or a bluebird.

D. Explainability

Explainability of machine learning is often touted as the missing piece complementing other assurance practices by providing confidence that the system is operating as intended or at least in a safe fashion. Complicating this argument is the fact that explainability means different things to different people meaning you always have to ask "explain what and to who"?

The "black box", common in aviation, is a well established engineering artifact allowing experts to determine the cause of accidents and incidents after the fact. Given that machine learning may not always react to new situations in predictable ways and in the worst case can endanger the public safety, a similar recording device that provides engineers with enough visibility to ascertain why given certain inputs the machine learning system behaved the way it did can provide valuable forensics evidence when an accident or incident occurs [39]. Although deploying such a black box is sound engineering practice and the data could be used to improve the system performance, this notion of explainability does not really improve the assurance processes.

A second notion of explainability exposes technical details to developers for the purpose of debugging. The internal operation as well as details about input data are presented to developers with expert knowledge during testing to help them understand why the system may not be performing as expected. While this helps the developers improve the quality of the product, it is not clear how it impacts assurance.

Another concept of explainability is targeted at users. Say an autonomous vehicle is driving down the highway and suddenly makes several lane changes and then exits the highway. There might be an innocent explanation, say, the vehicle detects a pothole and then is notified of an accident further down the highway. On the other hand, the machine learning system may not be performing in a desired manner. This notion of explainability aims to tell users what is going on using language they can understand. Work toward this goal remains in the realm of basic research and the hope is to eventually produce machine learning systems that can explain to users why they behave the way they do. The idea is to improve trust in machine learning enabled systems, but currently the given explanations are simply too technical for non-experts. Moreover, it is well known trust is

often misplaced, for safety-critical systems it is more important for the system to be trustworthy.

E. Licensing

In conventional safety-critical systems there is usually a clear dividing line between automation and human operator. The computing hardware and software undergo certification while the human operator is required to be licensed. Machine learning is often employed in autonomous systems to replace functions that have traditionally been carried out by humans. There have been a number of proposals to license the machine learning enabled components of a system that are replacing a human in an autonomous system. The idea seems reasonable as there is often a set of well documented skills and procedures that are expected to be mastered and demonstrated on the licensing exam. Yet in addition, there are often minimum age requirements intended to ensure a level of maturity. More research is needed to understand what life experiences contribute to being "mature enough" and how they factor into handling off-nominal situations. There is often mandatory apprenticeship or mentoring requirements that can sometimes last years with candidates who cannot demonstrate an ability to handle themselves in critical and seemingly chaotic situations washing out of the program. How to incorporate the human life experiences and general maturity into a machine learning system is difficult when we do not adequately understand what this means. Similarly we do not have a good understanding of the role an apprenticeship and mentor evaluation often plays in the license process. Although a very valuable subject for research, a number of complex questions need to be sufficiently addressed before licensing AI safety could play the same role it does in licensing humans.

V. A Path Forward

We have established that the key feature enabling the assurance of safety-critical systems is possessing actionable specifications. It is critical not to abandon this pillar of assuring safe systems for the sake of expediency. Instead, we should focus on building those systems for which we possess actionable specifications.

Within the domain of cyber-physical systems and aerospace in particular, there are a significant number of problems where machine learning can be applied to applications possessing actionable specifications [40]. For instance, ACAS Xu [41] is an aircraft collision avoidance system that not only sounds an alert when a potential collision is detected, but gives horizontal and vertical maneuver guidance. We know how to construct collision avoidance systems via conventional means and hence can construct actionable specifications that can be used to construct test oracles or properties to check at runtime. In some cases, we may not know how to obtain an actionable specification of a machine learning component itself, but can formulate an actionable specification to ensure safe operation based on other components of the system [40].

In autonomous systems, the most compelling use cases for machine learning are areas such as perception where there are no effective conventional solutions and where the only form of specification is a large data set. Although we currently do not know how to extract an actionable specification from such a data set, there is promising research. Mathematicians working in the area of topological data analysis [42]–[44] are bringing to bear powerful techniques from algebraic and differential topology as well as differential and algebraic geometry that allow us to compute geometric and topological invariants on the data. Persistent homology and distributed persistent homology [45] have emerged as techniques that help distinguish between actual features of high-dimensional data sets from noise in the data. Although the field is relatively new, the techniques have provided valuable insight into the differences between deep and shallow neural networks [46]. Given that in machine learning the data is the algorithm, it would seem that data invariants should be respected by the implementation and thus constitute actionable specifications that can be checked by testing, formal methods or at runtime. This is promising and exciting basic research being supported by NASA and other organizations, and will likely require some time to achieve a technical readiness level to transfer to industrial practice, but at present it appears to be our best hope for obtaining actionable specifications from large data sets.

VI. Conclusion

We have discussed how safety-critical systems are designed and assured to ensure the protection of the public safety and seen how machine learning poses a challenge to traditional design methodology. We have shown that when there are actionable specifications, it is possible to adapt established approaches to assure these systems. On the other hand, domains for which the only specification is a large data set remain a challenge. While topological data analysis shows promise to produce

actionable specifications from large data sets much basic research remains to be done. Over seventy years ago, Vannevar Bush authored his famous Frontiers of Science report [47] that laid out the course from basic science to applied science to industrial products. While there is always a temptation on the part of leaders to demand breakthroughs based on a schedule, this is one of those cases where the processes must run their course.

REFERENCES

[1] John C. Knight, "Safety critical systems: Challenges and directions," in *Proceedings of the 24th International Conference on Software Engineering*, ser. ICSE '02. ACM, 2002, pp. 547–550.

[2] Nancy G. Leveson, *Engineering a Safer World: Systems Thinking Applied to Safety*. MIT Press, 2012.

[3] P. Laplante, D. Milojicic, S. Serebryakov, and D. Bennett, "Artificial intelligence and critical systems: From hype to reality," *Computer*, vol. 53, no. 11, pp. 45–52, 2020.

[4] A. P. Members, "AFE 87 - Machine Learning," Aerospace Vehicle Systems Institute, Tech. Rep., 2020.

[5] D. Amodei, C. Olah, J. Steinhardt, P. F. Christiano, J. Schulman, and D. Mané, "Concrete problems in ai safety," *ArXiv*, vol. abs/1606.06565, 2016. [Online]. Available: https://arxiv.org/pdf/1606.06565.pdf

[6] R. McAllister, Y. Gal, A. Kendall, M. Van Der Wilk, A. Shah, R. Cipolla, and A. Weller, "Concrete problems for autonomous vehicle safety: Advantages of bayesian deep learning," in *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, ser. IJCAI'17. AAAI Press, 2017, pp. 4745–4753.

[7] J. Faria, "Machine learning safety: An overview," in *Proceedings of the Safety-critical Systems Symposium 2018 (SSS'18)*, 02 2018.

[8] K. R. Varshney, "Engineering safety in machine learning," 2016.

[9] N. R. Council, *Autonomy Research for Civil Aviation: Toward a New Era of Flight*. The National Academies Press, 2014.

[10] "Ansi/ul 4600: Standard for safety for the evaluation of autonomous products." [Online]. Available: https://ulse.org/UL4600

[11] SAE International, "Guidelines and methods for conducting the safety assessment process on civil airborne systems and equipment," SAE International, 1996, aRP 4761.

[12] ——, "Guidelines For Development Of Civil Aircraft and Systems," SAE International, 2010, aRP4754A.

[13] RTCA, "Software considerations in airborne systems and equipment certification," RTCA, Inc., 2011, RCTA/DO-178C.

[14] MIRA Ltd, *MISRA-C:2004 Guidelines for the use of the C language in Critical Systems*, MIRA Std., 2004.

[15] K. J. Hayhurst, D. S. Veerhusen, J. J. Chilenski, and L. K. Rierson, "A practical tutorial on modified condition/ decision coverage," NASA, Tech. Rep. NASA/TM-2001-210876, 2001.

[16] "Formal methods supplement to do-178c and do-278a," RTCA, Inc., 2011, RCTA/DO333.

[17] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[18] D. Su, H. Zhang, H. Chen, J. Yi, P.-Y. Chen, and Y. Gao, "Is robustness the cost of accuracy? – a comprehensive study on the robustness of 18 deep image classification models," 2018.

[19] G. Katz, C. Barrett, D. L. Dill, K. Julian, and M. J. Kochenderfer, "Towards proving the adversarial robustness of deep neural networks," in Proceedings First Workshop on *Formal Verification of Autonomous Vehicles,* Turin, Italy, 19th September 2017, ser. Electronic Proceedings in Theoretical Computer Science, L. Bulwahn, M. Kamali, and S. Linker, Eds., vol. 257. Open Publishing Association, 2017, pp. 19–26.

[20] Dusica Marijan and Arnaud Gotlieb and Kumar Ahuja Mohit, "Challenges of Testing Machine Learning Based Systems." San Francisco, CA, USA: IEEE, 2019.

[21] K. Pei, Y. Cao, J. Yang, and S. Jana, "Deepxplore: automated whitebox testing of deep learning systems," *Commun. ACM*, vol. 62, no. 11, pp. 137–145, 2019.

[22] Y. Sun, X. Huang, D. Kroening, J. Sharp, M. Hill, and R. Ashmore, "Deepconcolic: testing and debugging deep neural networks," in *Proceedings of the 41st International Conference on Software Engineering: Companion Proceedings, ICSE 2019, Montreal, QC, Canada, May 25-31, 2019*, J. M. Atlee, T. Bultan, and J. Whittle, Eds. IEEE / ACM, 2019, pp. 111–114.

[23] C. Liu, T. Arnon, C. Lazarus, C. Barrett, and M. J. Kochenderfer, "Algorithms for verifying deep neural networks," 2019. [Online]. Available: https://arxiv.org/abs/1903.06758

[24] R. Bunel, I. Turkaslan, P. H. Torr, P. Kohli, and M. P. Kumar, "Piecewise linear neural networks verification: A comparative study," 2018. [Online]. Available: https://openreview.net/forum?id=BkPrDFgR-

[25] M. Akintunde, A. Lomuscio, L. Maganti, and E. Pirovano, "Reachability analysis for neural agent-environment systems," in *Principles of Knowledge Representation and Reasoning: Proceedings of the Sixteenth International Conference, KR 2018, Tempe, Arizona, 30 October - 2 November 2018*, M. Thielscher, F. Toni, and F. Wolter, Eds. AAAI Press, 2018, pp. 184–193.

[26] L. de Moura, B. Dutertre, and N. Shankar, "A tutorial on satisfiability modulo theories," in *Computer Aided Verification*, W. Damm and H. Hermanns, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 20–36.

[27] G. Katz, C. Barrett, D. L. Dill, K. Julian, and M. J. Kochenderfer, "Reluplex: An efficient SMT solver for verifying deep neural networks," in *Proceedings of the 29th International Conference on Computer Aided Verification (CAV '17)*, ser. Lecture Notes in Computer Science, R. Majumdar and V. Kuncak, Eds., vol. 10426, no. 1. Springer, Jul. 2017, pp. 97–117, heidelberg, Germany. [Online]. Available: http://www.cs.stanford.edu/~barrett/pubs/KBD+17.pdf

[28] G. Katz, D. A. Huang, D. Ibeling, K. Julian, C. Lazarus, R. Lim, P. Shah, S. Thakoor, H. Wu, A. Zeljić, D. L. Dill, M. J. Kochenderfer, and C. Barrett, "The marabou framework for verification and analysis of deep neural networks," in *Proceedings of the 31st International Conference on Computer Aided Verification (CAV '19)*, ser. Lecture Notes in Computer Science, I. Dillig and S. Tasiran, Eds., vol. 11561. Springer International Publishing, 2019, pp. 443–452, new York, New York. [Online]. Available: http://www.cs.stanford.edu/~barrett/pubs/KHI+19.pdf

[29] P. Cousot and R. Cousot, "Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints." in *POPL '77: Proceedings of the 4th ACM SIGACT-SIGPLAN symposium on Principles of programming languages*. ACM Press, 1977, pp. 238–252.

[30] T. Gehr, M. Mirman, D. Drachsler-Cohen, P. Tsankov, S. Chaudhuri, and M. Vechev, "AI^2: Safety and robustness certification of neural networks with abstract interpretation," in *Security and Privacy (SP), 2018 IEEE Symposium on*, 2018.

[31] G. Singh, T. Gehr, M. Püschel, and M. Vechev, "An abstract domain for certifying neural networks," *Proc. ACM Program. Lang.*, vol. 3, no. POPL, pp. 41:1–41:30, Jan. 2019.

[32] A. Goodloe and L. Pike, "Monitoring distributed real-time systems: A survey and future directions," NASA Langley Research Center, Tech. Rep. NASA/CR-2010-216724, July 2010.

[33] E. Bartocci, Y. Falcone, A. Francalanza, and G. Reger, "Introduction to runtime verification," in *Lectures on Runtime Verification - Introductory and Advanced Topics*, ser. Lecture Notes in Computer Science, E. Bartocci and Y. Falcone, Eds. Springer, 2018, vol. 10457, pp. 1–33.

[34] Y. Falcone, K. Havelund, and G. Reger, "A tutorial on runtime verification," in *Engineering Dependable Software Systems*, ser. NATO Science for Peace and Security Series, D: Information and Communication Security, M. Broy, D. A. Peled, and G. Kalus, Eds. IOS Press, 2013, vol. 34, pp. 141–175.

[35] "Copilot," https://copilot-language.github.io, Accessed Aug 01, 2021.

[36] I. Perez, F. Dedden, and A. Goodloe, "Copilot 3," NASA Langley Research Center, Tech. Rep. NASA/TM–2020–220587, April 2020.

[37] A. Goodloe, "Challenges in high-assurance runtime verification," in *Leveraging Applications of Formal Methods, Verification and Validation: Foundational Techniques - 7th International Symposium, ISoLA 2016, Imperial, Corfu, Greece, October 10-14, 2016, Proceedings, Part I*, 2016, pp. 446–460.

[38] I. Perez, A. Mavridou, T. Pressburger, A. Goodloe, and D. Giannakopoulou, "Automated translation of natural language requirements to runtime monitors," in *Tools and Algorithms for the Construction and Analysis of Systems - 28th International Conference, TACAS 2022, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2022, Munich, Germany, April 2-7, 2022, Proceedings, Part I*, ser. Lecture Notes in Computer Science, D. Fisman and G. Rosu, Eds., vol. 13243. Springer, 2022, pp. 387–395.

[39] G. Falco, B. Shneiderman, J. Badger, R. Carrier, A. Dahbura, D. Danks, M. Eling, A. Goodloe, J. Gupta, C. Hart, M. Jirotka, H. Johnson, C. Lapointe, A. J. Llorens, A. K. Mackworth, C. Maple, S. E. Pálsson, F. Pasquale, A. F. T. Winfield, and Z. K. Yeong, "Governing AI safety through independent audits," *Nat. Mach. Intell.*, vol. 3, no. 7, pp. 566–571, 2021.

[40] D. Cofer, I. Amundson, R. Sattigeri, A. P. C. Boggs, E. Smith, L. Gilham, T. Byun, and S. Rayadurgam, "Run-time assurance for learning-enabled systems," in *NASA Formal Methods: 12th International Symposium, NFM 2020, Moffett Field, CA, USA, May 11–15, 2020, Proceedings*. Berlin, Heidelberg: Springer-Verlag, 2020, p. 361–368.

[41] M. P. Owen, A. Panken, R. Moss, L. Alvarez, and C. Leeper, "ACAS Xu: Integrated Collision Avoidance and Detect and Avoid Capability for UAS," in *2019 IEEE/AIAA 38th Digital Avionics Systems Conference (DASC)*, 2019, pp. 1–10.

[42] H. Edelsbrunner and J. Harr, *Computational Topology:An Introduction*. AMS Press, 2010.

[43] F. Chazal and B. Michel, "An introduction to topological data analysis: Fundamental and practical aspects for data scientists," *Frontiers in Artificial Intelligence*, vol. 4, 2021. [Online]. Available: https://www.frontiersin.org/articles/10.3389/frai.2021.667963

[44] G. Carlsson, "Topology and data," *Bulletin of The American Mathematical Society - BULL AMER MATH SOC*, vol. 46, pp. 255–308, 04 2009.
in *38th International Symposium on Computational Geometry (SoCG 2022)*, ser. Leibniz International Proceedings in Informatics (LIPIcs), X. Goaoc and M. Kerber, Eds., vol. 224. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022, pp. 61:1–61:16.

[45] E. Solomon, A. Wagner, and P. Bendich, "From Geometry to Topology: Inverse Theorems for Distributed Persistence,"

[46] G. Naitzat, A. Zhitnikov, and L.-H. Lim, "Topology of deep neural networks," 2020. [Online]. Available: https://arxiv.org/abs/2004.06093

[47] V. Bush, "Science–the endless frontier : a report to the president on a program for postwar scientific research." National Science Foundation, 1980.