International Journal on Software Tools for Technology Transfer (2023) 25:241–247
https://doi.org/10.1007/s10009-023-00715-0

Introduction

Explanation Paradigms Leveraging Analytic Intuition (ExPLAIn)

Nils Jansen[1] · Gerrit Nolte[2] · Bernhard Steffen[2]

Accepted: 3 July 2023 / Published online: 1 September 2023

Abstract
In this paper, we present the envisioned style and scope of the new topic "Explanation Paradigms Leveraging Analytic Intuition" (ExPLAIn) with the International Journal on Software Tools for Technology Transfer (STTT). Intention behind this new topic is to (1) explicitly address all aspects and issues that arise when trying to, if possible, reveal and then confirm hidden properties of black-box systems, or (2) to enforce vital properties by embedding them into appropriate system contexts. Machine-learned systems, such as Deep Neural Networks, are particularly challenging black-box systems, and there is a wealth of formal methods for analysis and verification waiting to be adapted and applied. The selection of papers of this first Special Section of ExPLAIn, most of which were co-authored by editorial board members, is an illustrative example of the style and scope envisioned: In addition to methodological papers on verification, explanation, and their scalability, case studies, tool papers, literature reviews, and position papers are also welcome.

Keywords Formal methods · Explanation · (Deep) Neural Networks · Deep learning · Rigorous analysis · Decision trees · Random forests · Robustness · Reliability · Verification · (Statistical) model checking · Testing · Competition

1 Introduction

Today's decision-making is increasingly aided by computers, be it in medicine, the stock market, or when we follow recommendations provided by various platforms. Few people question the automatic support systems and ask for explanations. In the majority of cases, this might be harmless, in particular when the damage caused by wrong results remains moderate. The situation changes, however, when critical decisions with a potential for disaster are delegated to potentially unsafe systems.

This situation is, of course, not novel. The adoption of software systems into safety-critical domains has been a topic for decades, and there is a wealth of methods and tools for testing, verifying, and explaining software to ensure its safety, e.g., for industrial critical systems [1, 7, 36]. Recent advancements in computer science, in particular in the realm of machine learning, force us to radically rethink the possible and the desirable: How can we exploit the huge new potential and what price are we prepared to pay.

We are only at the very beginning of understanding the power and impact of "Ecorithms" as presented by Leslie Valiant [43]. They provide a new computational paradigm based on learning from observation/examples (rather than conceptual design), which Valiant therefore characterizes as "theoryless". The power of ecorithms, which largely escape human control, becomes particular apparent with today's large language models. Valiant recognized the importance of ecorithms very early on. In fact, he conjectured already a decade ago that they may even serve as a new paradigm for explaining the process of evolution and, in particular, its short timeframe.

It is a major challenge to establish a bridge between the "theoryless" and the "theoryful" [43], i.e., the (traditional) computational paradigm we are used to, trust, and feel comfortable with. On the other hand, this bridge is essential to responsibly include ecorithms in safety-critical solutions for, e.g., automotive driving.

There exists a wealth of methods and tools for dealing with theoryful algorithms and applications that the formal methods community has established in recent decades. ExPLAIn

✉ B. Steffen
steffen@cs.tu-dortmund.de

N. Jansen
n.jansen@science.ru.nl

G. Nolte
gerrit.nolte@tu-dortmund.de

1 Radboud University Nijmegen, Nijmegen, The Netherlands

2 TU Dortmund, Dortmund, Germany

invites contributions that exploit and adapt this wealth to achieve a synergy between the theoryless and the theoryful that justifies the responsible use of corresponding solutions also for safety-critical systems.

The remainder of this paper aims at indicating what corresponding contributions might look like.

2 Challenges for the responsible use of machine learning

Perhaps no field of computer science exemplifies the trend of evolving software systems more than artificial intelligence and, in particular, machine learning. Driven by the abundance of data and the influx of strong computing hardware, machine learning has risen to an immense level of prominence in the realm of computer science during recent decades. From language processing [9] and computer vision [14, 41] to playing complex games [40] and self-driving cars [37], machine learning has found success in domains where it is almost impossible to succeed with traditional, handwritten programs. Moreover, machine learning promises not only solutions to previously hard problems, but also to achieve them without the need for direct human input. Today, even incredibly hard problems can be solved automatically by machine learning systems, replacing hard human work and intuition with large datasets and computing power.

However, as much as the success stories of deep learning motivate its increasing adoption in real life, the metaphorical coins' flipside is equally hard to overcome. While the dream of automated, high-performance software is incredibly appealing, it necessarily comes with substantial drawbacks concerning the safety and reliability of the systems at hand. Unlike traditional, handwritten programs, machine learning systems are not the result of human consideration, but rather of a training process where the system is confronted with a large number of examples. While machine learning systems are typically able to perform correctly in situations that were contained in these examples, their behavior in novel, unseen situations is largely uncontrolled. Moreover, biases in datasets might lead to disastrous consequences even if situations have been encountered before.

Issues prominently arise with respect to both algorithmic fairness [11, 29], where machine learning systems reinforce existing social biases, and safety concerns [3, 44], where machine learning systems are not yet reliable enough to be trusted with safety-critical tasks, a well-known example being car crashes involving self-driving cars [25]. For machine learning systems to be widely employed in everyday applications, this status quo needs to change. Systems that operate on a scale that impacts human lives in any meaningful way should be reliable, fair, safe, comprehensible, and trustworthy to the people that it affects.

We need to ensure the responsible use of machine learning in real-world applications.

Safety concerns in machine learning are conceptually reminiscent of the situation originally encountered by the formal methods community. The field of formal methods set out to verify and explain complex software and hardware systems using mathematically rigorous methods that ensure the reliability of such systems, even in extremely safety-critical domains. As an example, formal methods-based automated approaches to software/system verification aim at provable correctness, which is particularly important in the context of industrial critical systems.

In both areas (machine learning and formal methods), system understanding aims at explaining the properties of an unfeasible (black box) system in a feasible (white box) form. In some way, the results of formal methods are often those that are currently lacking from the status quo of machine learning research.

A major challenge is to adapt and tailor the existing approaches and ideas from formal methods to the new domain of machine learning. Bridging this gap is by no means trivial. Machine learning systems come with a distinct profile for verification and explanation tasks that are specifically tailored to this profile.

This leads us to a core challenge of machine learning, relating to systems that are rigorously verified and whose inner workings are explainable to the involved stakeholders.

How can formal methods help to enable the responsible use of machine learning?

3 Challenges in formal methods

Since its inception, the field of formal methods has played a critical role in the adoption of software systems into critical domains. From program analysis to model checking and automata learning, formal methods offer avenues to prevent critical system errors in a reliable, mathematically rigorous manner.

Precisely this rigor and reliability of formal methods have facilitated their adoption in a variety of safety-critical industrial domains, spanning a wide range from aerospace applications to healthcare systems. As put by Edmund Clarke [10]:

> The use of formal methods does not a priori guarantee correctness. However, they can greatly increase our understanding of a system by revealing inconsistencies, ambiguities, and incompleteness that might otherwise go undetected

Despite this, formal methods often face a natural roadblock. Many of the objects considered by formal methods are, by nature, too complex to rigorously verify and/or explain. For example, many variants of model checking and program verification are known to be theoretically hard or even undecidable [6, 39]. Thus, scaling the formal methods approach to industrial scope systems poses a critical challenge.

Machine learning, on the other hand, strives to find efficient solutions in complex domains and provides numerous approximate, but efficient algorithms that use probabilistic approaches to ensure scalability. Thus, while machine learning approaches are scalable, they miss the rigor and reliability that are core to formal methods. However, to profit from formal methods in machine learning, their different profiles need to be aligned.

The core challenge here can be formulated as such:

How can machine learning methods and formal methods solve large-scale problems in tandem to ensure mathematical rigor and precision in a scalable fashion?

4 Goals of ExPLAIn

As outlined in the previous sections, the fields of formal methods and machine learning can benefit greatly t from synergistic approaches despite, or perhaps because of, their drastically different profiles.

The track "Explanation Paradigms Leveraging Analytic Intuition", or "ExPLAIn" for short, seeks to connect these fields and give an avenue for researchers to present work that concerns the explanation, verification, and more generally, safety and the responsible use of complex software systems, with an emphasis on systems with a black-box character. Machine learned systems, such as Deep Neural Networks, are particularly challenging black-box systems, and there is a wealth of formal methods for analysis and verification waiting to be adapted and applied.

ExPLAIn therefore addresses researchers with background in machine learning, general artificial intelligence, or formal methods with an interest to investigate the interface between these domains, which have recently been identified as particular views on neurosymblic AI, where formal methods take the role of an extended version of symbolic AI [20, 31].

The selection of papers of this initial Special Section of ExPLAIn, most of which were co-authored by editorial board members, is an illustrative example for the style and scope envisioned. They fall into five categories:

- Approaches leveraging formal methods to verify machine learning systems in an analytically rigorous way.
- Approaches leveraging formal methods to test machine learning systems in a structural way yielding either probabilistic guarantees or guarantees on behavior coverage.
- Approaches leveraging formal methods to explain opaque machine learning systems by transforming them into white-box systems that are semantically equivalent.
- Approaches incorporating probabilistic techniques from machine learning in formal methods while maintaining precision or statistically bounding the error arising from the use of probabilistic methods.
- Approaches using techniques from machine learning as heuristics in formal methods to achieve domain-specific scalability.

ExPLAIn is, however, not restricted to these categories: In addition to methodological papers, case studies, tool papers, literature reviews, industrial experiences, and position papers are also welcome.

5 ExPLAIn's first edition

5.1 Explaining machine learning systems

Explaining machine learning systems means rendering the inner workings of a machine learning system understandable to an end user. This task is very unrestrictive: In principle, any approach that either explains a machine learning system (model explanations) or its decisions (outcome explanation) in any fashion that is comprehensible to a human is desirable.

Most often, this entails distilling the complex and opaque machine learning system into an equivalent simple, easy-to-comprehend system. There exist two flavors of this: Post-hoc explanations that transform already existing machine learning systems into more comprehensible forms, and "self-explaining" machine learning models that are trained such that they are by virtue of their construction more comprehensible. Especially interesting in this regard are decision trees as both expressive and structurally comprehensible models and therefore a perfect fit for the profile of a surrogate, explanation model. The following papers use decision trees to achieve explainable machine learning systems.

Decision trees and SVMs In their paper "Analytically Explainable Controllers: Decision Trees and Support Vector Machines Join Forces" [21], the authors seek to combine decision trees [12] and support vector machines [34] to achieve "self-explaining" controllers for hybrid systems, i.e., systems where discrete and continuous variables interact.

Structurally, decision trees are commonly accepted as comprehensible machine learning models [2], provided that they are not prohibitively large and each individual predicate occurring in the tree is comprehensible. The challenge, then, is to find predicates that are both descriptive enough for a small decision tree to perform well and simple enough to be individually comprehensible.

As the authors argue, the linear predicates that are customary in most decision tree implementations are insufficient to achieve this goal. Instead, as domains become more complex, more advanced, non-linear predicates are required. The authors use support vector machines (SVMs) [34] to find good separators of the data points and use these SVMs as predicates in a decision tree. By introducing a non-linear transform into the process (akin to the kernel trick that is common to SVMs), the authors attain non-linear predicates, allowing for much more expressive predicates. In their experiments, the authors show that this allows the construction of small, easy-to-understand decision trees.

The following papers concern post-hoc explanations of various machine learning systems, ranging from random forests to neural networks:

Aggregating random forests Random forests consist of multiple decision trees that work in an ensemble, making decisions by a majority vote [13]. As noted before, decision trees are widely accepted as structurally explainable machine learning systems. Random forests, however, feature a multitude of trees that are evaluated in parallel. This parallel structure is much harder for humans to follow and therefore poses a challenge with regards to explainability. In the paper "Algebraic Aggregation of Random Forests: Towards Explainability and Rapid Evaluation" [17], the authors transform each tree in a random forest into a semantically equivalent algebraic decision diagram (ADD) [5] and leverage their algebraic properties to merge the trees into one singular ADD that faithfully captures the semantics of the entire forest. Structurally, the resulting ADD is equivalent to a single decision structure and uses only predicates from each singular tree, making it a comprehensible model.

Moreover, using the algebraic properties of ADDs, the authors also show how this ADD can be used to explain singular outcomes and characterize inputs belonging to a given class. The authors also introduce infeasible path reductions by removing redundant predicates and paths in an ADD that no input can satisfy. They also show experimentally that the resulting ADDs can be quite small and explainable.

Aggregating random forests - a webtool Building on this approach, the paper "Forest GUMP: A Tool for Verification and Explanation" [33] presents Forest GUMP (short for: General, Unifying Merge Process), a web-based tool implementing the algebraic aggregation of random forests as described in [17]. The paper includes in-depth, practical descriptions of the tool and illustrates its interaction with the user. Moreover, the paper includes an experimental evaluation of the approach, showing that by removing semantically infeasible paths from ADDs, the resulting structures can be much more concise. In a more theoretical aspect, the paper also shows how pre-/post-condition verification of random forests can be implemented using its aggregating technology. Further, the paper also provides a method to decide whether two random forests are identical and to provide evidence in case they are not.

Transforming neural networks to decision trees Neural networks are one of the hallmark models of modern machine learning. Responsible for many of machine learning's largest success stories in the recent decade [9, 16, 45], neural networks are also infamous for their opaque, black-box nature. The reason for this lies in their complex structure, consisting of both highly parallel and non-linear operations that quickly exceed human capabilities. In their paper "Towards Rigorous Understanding of Neural Networks via Semantics-preserving Transformations" [38], they propose a novel method combining ideas from ADDs and symbolic execution [24] to transform ReLU neural networks into semantically equivalent decision tree-like structures called Typed Affine Decision Structures (TADSs). Much like ADDs, TADSs support a variety of algebraic operations, including scalar multiplication, addition, subtraction, and equality checks. The authors show how these algebraic properties can be used to elegantly answer many interesting questions regarding neural network verification and decide the full equivalence or approximate equivalence of two neural networks.

5.2 Verification

Program verification has a long history in computer science. In many safety-critical domains, software engineers are required to prove that the systems they provide adhere to certain safety restrictions, i.e., proving that certain safety properties always hold or that certain unsafe states are never reached. This applies perhaps even more so to the infamously hard-to-comprehend machine learning systems. Only if a black-box system can be proven to be safe is it fit for use in safety-critical domains.

The following papers present advances in the field of neural network verification.

Verification of neural networks competition As neural network verification advances, the field widens. At present, a multitude of different neural network verifiers exist, each employing different approaches and techniques. The Verification of Neural Networks Competition (VNN-COMP) aims to evaluate these solvers on a suite of benchmarks consisting of neural networks trained on practical problems. After the first three iterations of this competition, its organizers give an overview over its results and the current status quo in neural network verification in their paper "First Three Years of the International Verification of Neural Networks Competition (VNN-COMP)" [8]. They discuss the goals and results of this competition and the possible next steps to further develop the competition setting. Further, they discuss the winning entries of this challenge, identify trends in modern neural network verification, and examine the year-to-year development in the field.

Property directed verification of recurrent neural networks Recurrent neural networks operate on time series, taking one input after another, usually maintaining some sort of memory of the previously seen inputs. Verifying a recurrent neural network end-to-end with standard methods would involve verifying the unrolled network, which is usually prohibitively large. The authors of the paper "Analysis of Recurrent Neural Networks via Property-Directed Verification of Surrogate Models" [23] propose a property-directed method that is specifically tailored to the verification of recurrent neural networks. They do so by using automata learning to iteratively refine an automaton that represents the classification behavior of the neural network under consideration. At each step, they use the intermediate automaton to attain potential candidates for safety violations and use these candidates to either disprove the safety property under consideration or refine the intermediate automaton.

Neural network verification with TADS While Typed Affine Decision Structures (TADSs) are introduced as white-box models for the precise explanation of neural networks, the paper "The Power of Typed Affine Decision Structures: A Case Study" [35] discusses their application to the verification of local properties of neural networks. To this end, the authors extend TADSs by preconditions, allowing TADSs to focus only on a specific input region that is characterized by a precondition, and the argmax function, which is typically used in neural network classifiers. The authors show that, in principle, TADSs can be used to verify neural networks in a very conceptually simple fashion, but they suffer from scalability issues. To remedy this, the authors propose the usage of the popular dimensionality reduction method PCA to achieve easier scalability and prove that it can be safely used to prove local properties of neural networks.

Safe and robust decision making under uncertainty One of the premier avenues of machine learning is the task of making decisions in uncertain scenarios. The idea of uncertain systems and unknown scenarios is very applicable to real-world applications, where processes are usually not fully understood and involve some measure of randomness. Making decisions in an uncertain world is a challenging and important question.

This uncertainty about the world can come in two flavors: Aleatoric uncertainty, which is based on an actual random, uncertain process in a given system, and epistemic uncertainty, which is not inherent to the system but arises from a lack of knowledge about the existing processes. Epistemic uncertainty can be reduced by gathering additional information about a given system; Aleatoric uncertainty is a natural constant of a given system and cannot be reduced. In real-world applications, this is an important distinction, and adequately treating uncertainty is critical.

In their paper "Decision-making under uncertainty: beyond probabilities. Challenges and Perspectices" [4], the authors give an introductory overview of the field of decision making under uncertainty with a special focus on the different ways in which aleatoric and epistemic uncertainty are mathematically formalized. Furthermore, the authors introduce robust reinforcement learning [32], which seeks to not only make good decisions, but decisions that are in some respect safe, and bayesian reinforcement learning [15, 26], which provides a natural mechanism for the modeling of epistemic uncertainty in bayesian prior distributions. Lastly, the authors present current challenges in the field and motivate their importance.

5.3 Structured testing

Much like in traditional software engineering, verification is usually quite costly. In domains where safety is not as critical, testing therefore becomes much more attractive. The following papers present different methods for the structured testing of machine learning systems, involving either coverage criteria, guaranteeing that the behavior of a given system is aptly covered, or statistical methods that can guarantee correctness up to a given probability.

Neural network testing coverage In traditional programming, there exist clear notion of test coverage, indicating whether a given test suite covers the potential behaviors of a program sufficiently enough to inspire trust in the program [30]. In neural network testing, similar coverage methods are in use. Typical examples include covering potential activation patterns or ensuring that all neurons in a network are covered [28]. DNNCov, an integrated framework for the testing neural network models, aims to provide users with coverage information about a given neural network and training set. An introduction to the tool and the coverage metrics it supports is given in the paper "An Overview of Structural Coverage Metrics for Testing Neural Networks" [42]. Moreover, the authors discuss existing coverage metrics and use DNNCov to examine their practical implications, namely, whether a higher coverage implies a higher probability of finding potential errors, revealing that existing coverage metrics might be insufficient to aptly cover neural networks.

Statistical model checking for deep reinforcement learning Reinforcement learning (RL) entails agents learning by themselves in an unfamiliar world, both exploring the world and learning to act within it [22]. As a result, verifying RL agents requires verifying the agents behavior *in context* of the environment in which it acts, leading to complex verification tasks. A light-weight, approximate alternative to the verification of RL agents is Deep Statistical Model Checking (DSMC), as proposed in the paper "Analyzing Neural Network Behavior through Deep Statistical Model Checking" [18]. As the authors show, an RL agent acting on a markov decision process, the standard formulation of an environment in RL, can be modeled as a Markov chain, motivating the use of probabilistic model checking to verify its safety [19]. Probabilistic model checking is exact, yet expensive. The titular statistical model checking provides a light-weight approximate method for verification based on simulated runs of the system [27], yielding DSMC, an efficient, approximate method for verifying safety of neural network agents in an RL setting. The authors provide an experimental evaluation on the racetrack benchmark problem, showing that deep statistical model checking can provide valuable information about safety-critical behavior.

Acknowledgements As editors of the ExPLAIn theme, we would like to thank all authors for their contributions. We would also like to thank all reviewers for their insights and helpful comments.

Funding Open Access funding enabled and organized by Projekt DEAL.

References

1. ERCIM working group on formal methods for industrial critical systems (FMICS). http://fmics.inria.fr/. Accessed: 2023-07-28
2. Adadi, A., Berrada, M.: Peeking inside the black-box: a survey on explainable artificial intelligence (xai). IEEE Access **6**, 52138–52160 (2018)
3. Amodei, D., Olah, C., Steinhardt, J., Christiano, P.F., Schulman, J., Mané, D.: Concrete problems in AI safety CoRR (2016). arXiv:1606.06565
4. Badings, T., Simao, T., Suilen, M., Jansen, N.: Decision-making under uncertainty: beyond probabilities. Challenges and perspectives. https://doi.org/10.1007/s10009-023-00704-3
5. Bahar, R.I., Frohm, E.A., Gaona, C.M., Hachtel, G.D., Macii, E., Pardo, A., Somenzi, F.: Algebric decision diagrams and their applications. Form. Methods Syst. Des. **10**(2), 171–206 (1997)
6. Baier, C., Katoen, J.P.: Principles of Model Checking. MIT Press, Cambridge (2008)
7. Blanchet, B., Cousot, P., Cousot, R., Feret, J., Mauborgne, L., Miné, A., Monniaux, D., Rival, X.: A static analyzer for large safety-critical software. In: Proceedings of the ACM SIGPLAN 2003 Conference on Programming Language Design and Implementation, pp. 196–207 (2003)
8. Brix, C., Müller, M., Bak, S., Johnson, T.T., Liu, C.: First three years of the international verification of neural networks competition (vnn-comp). https://doi.org/10.1007/s10009-023-00703-4
9. Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J.D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al.: Language models are few-shot learners. Adv. Neural Inf. Process. Syst. **33**, 1877–1901 (2020)
10. Clarke, E.M., Wing, J.M.: Formal methods: state of the art and future directions. ACM Comput. Surv. **28**(4), 626–643 (1996)
11. Corbett-Davies, S., Goel, S.: The measure and mismeasure of fairness: a critical review of fair machine learning (2018). ArXiv preprint. arXiv:1808.00023
12. Cramer, G., Ford, R., Hall, R.: Estimation of toxic hazard—a decision tree approach. Food Cosmet. Toxicol. **16**(3), 255–276 (1976)
13. Cutler, A., Cutler, D.R., Stevens, J.R.: Random forests. In: Ensemble Machine Learning, pp. 157–175. Springer, Berlin (2012)
14. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al.: An image is worth 16x16 words: Transformers for image recognition at scale (2020). ArXiv preprint. arXiv:2010.11929
15. Ghavamzadeh, M., Mannor, S., Pineau, J., Tamar, A., et al.: Bayesian reinforcement learning: a survey. Found. Trends Mach. Learn. **8**(5-6), 359–483 (2015)
16. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press, Cambridge (2016)
17. Gossen, F., Steffen, B.: Algebraic aggregation of random forests: towards explainability and rapid evaluation. Int. J. Softw. Tools Technol. Transf. (2021). https://doi.org/10.1007/s10009-021-00635-x
18. Gros, T.P., Hermanns, H., Hoffmann, J., Klauck, M., Steinmetz, M.: Analyzing neural network behavior through deep statistical model checking. https://doi.org/10.1007/s10009-022-00685-9
19. Hérault, T., Lassaigne, R., Magniette, F., Peyronnet, S.: Approximate probabilistic model checking. In: International Workshop on Verification, Model Checking, and Abstract Interpretation, pp. 73–84. Springer, Berlin (2004)
20. Jansen, N.: Intelligent and dependable decision-making under uncertainty. In: FM, Lecture Notes in Computer Science, vol. 14000, pp. 26–36. Springer, Berlin (2023)
21. Jüngermann, F., Křetínský, J., Weininger, M.: Algebraically explainable controllers: Decision trees and support vector machines join forces. Int. J. Softw. Tools Technol. Transf. (in press)
22. Kaelbling, L.P., Littman, M.L., Moore, A.W.: Reinforcement learning: a survey. J. Artif. Intell. Res. **4**, 237–285 (1996)
23. Khmelnitsky, I., Neider, D., Roy, R., Xie, X., Barbot, B., Bollig, B., Finkel, A., Haddad, S., Leucker, M., Ye, L.: Analysis of recurrent neural networks via property-directed verification of surrogate models. Int. J. Softw. Tools Technol. Transf. (2022, in press)
24. King, J.C.: Symbolic execution and program testing. Commun. ACM **19**(7), 385–394 (1976)
25. Kohli, P., Chadha, A.: Enabling pedestrian safety using computer vision techniques: a case study of the 2018 uber inc. self-driving car crash. In: Future of Information and Communication Conference, pp. 261–279. Springer, Berlin (2019)
26. Lee, P.M.: Bayesian Statistics. Oxford University Press, London (1989)
27. Legay, A., Delahaye, B., Bensalem, S.: Statistical model checking: an overview. In: International Conference on Runtime Verification, pp. 122–135. Springer, Berlin (2010)

28. Ma, L., Juefei-Xu, F., Zhang, F., Sun, J., Xue, M., Li, B., Chen, C., Su, T., Li, L., Liu, Y., et al.: Deepgauge: multi-granularity testing criteria for deep learning systems. In: Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering, pp. 120–131 (2018)
29. Mehrabi, N., Morstatter, F., Saxena, N., Lerman, K., Galstyan, A.: A survey on bias and fairness in machine learning. ACM Comput. Surv. **54**(6), 1–35 (2021)
30. Miller, J.C., Maloney, C.J.: Systematic mistake analysis of digital computer programs. Commun. ACM **6**(2), 58–63 (1963)
31. Monroe, D.: Neurosymbolic AI. Commun. ACM **65**(10), 11–13 (2022)
32. Morimoto, J., Doya, K.: Robust reinforcement learning. Neural Comput. **17**(2), 335–359 (2005)
33. Murtovi, A., Bainczyk, A., Nolte, G., Schlüter, M., Bernhard, S.: Forest Gump: a tool for veification and explanation. Int. J. Softw. Tools Technol. Transf. (2022, in press)
34. Noble, W.S.: What is a support vector machine? Nat. Biotechnol. **24**(12), 1565–1567 (2006)
35. Nolte, G., Schlüter, M., Murtovi, A., Steffen, B.: The power of typed affine decision structures: a case study. https://doi.org/10.1007/s10009-023-00701-6
36. Parnas, D.L., Van Schouwen, A.J., Kwan, S.P.: Evaluation of safety-critical software. Commun. ACM **33**(6), 636–648 (1990)
37. Rao, Q., Frtunikj, J.: Deep learning for self-driving cars: chances and challenges. In: Proceedings of the 1st International Workshop on Software Engineering for AI in Autonomous Systems, pp. 35–38 (2018)
38. Schlüter, M., Nolte, G., Steffen, B.: Towards rigorous understanding of neural networks via semantics preserving transformation. Int. J. Softw. Tools Technol. Transf. (2022, in press)
39. Sieber, K.: The Foundations of Program Verification. Springer, Berlin (2013)
40. Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al.: Mastering the game of go without human knowledge. Nature **550**(7676), 354–359 (2017)
41. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition (2014). ArXiv preprint. arXiv:1409.1556
42. Usman, M., Sun, Y., Gopinath, D., Dange, R., Manolache, L., Pasareanu, C.: An overview of structural coverage metrics for testing neural networks. Int. J. Softw. Tools Technol. Transf. (2022, in press)
43. Valiant, L.: Probably Approximately Correct: Nature's Algorithms for Learning and Prospering in a Complex World. Basic Books, USA (2013)
44. Varshney, K.R.: Engineering safety in machine learning. In: 2016 Information Theory and Applications Workshop (ITA), pp. 1–5 (2016). https://doi.org/10.1109/ITA.2016.7888195
45. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. Adv. Neural Inf. Process. Syst. **30** (2017)