Preprint of R. Bloomfield, H. Khlaaf, P. Ryan Conmy and G. Fletcher, "Disruptive Innovations and Disruptive Assurance: Assuring Machine Learning and Autonomy" in *Computer*, vol. 52, no. 09, pp. 82-89, 2019.
doi: 10.1109/MC.2019.2914775
keywords: {autonomous systems;safety;disruptive innovation;machine learning}
url: https://doi.ieeecomputersociety.org/10.1109/MC.2019.2914775

ADELARD

24 Waterside
44–48 Wharf Road
London
N1 7UX

T +44 20 7832 5850
F +44 20 7832 5870
E office@adelard.com
W www.adelard.com

Disruptive innovations and disruptive assurance: assuring ML and autonomy

Robin Bloomfield[1,2], Heidy Khlaaf[1], Philippa Ryan[1], Gareth Fletcher[1]

[1]Adelard LLP [2]City, University of London

Introduction

The advancement and adoption of Machine Learning (ML) algorithms are a crucial innovative disruption. However, to benefit from these innovations within security and safety-critical domains we need to be able to evaluate the risk and benefits of the technologies used: in particular we need to assure ML-based and autonomous systems.

The assurance of complex software-based systems often relies on a standards-based justification, but in the case of autonomous systems, it is difficult to rely solely on this approach given the lack of validated standards, policies, and guidance for such novel technologies. Also, other strategies such as "drive to safety" - using evidence developed from trials and experience to support claims of safety in deployment - is unlikely to be successful by itself, as noted by [6] [7] especially if the impact of security threats are taken into account. This reinforces the need for innovation in assurance and the development of an assurance methodology for autonomous systems.

Although forthcoming standards and guidelines will eventually have an important, yet indirect role in helping us justify behaviours, we need further development of assurance frameworks to enable us to exploit disruptive technologies. We focus on directly investigating the desired behaviour (e.g., safety property or reliability) of a system through an argument or outcome-based approach that integrates disparate sources of evidence whether from compliance, experience or product analysis. We argue that building trust and trustworthiness through argument-based mechanisms, specifically, the Claims, Arguments, and Evidence (CAE) framework (see Table 1), allows for the accelerated exploration of novel mechanisms that would lead to the quality advancement and assurance of disruptive technologies.

The CAE framework supports the creates structured argumentation for complex engineering systems. It is based on an explicit claim-based approach to justification, and relates back to earlier philosophical work Toulmin and Wigmore as well as drawing on theory and empirical research in recent years in the safety and assurance cases areas (see John Rushby's analysis [4] for a rigorous review of the field

At the heart of the CAE framework are three key elements. Claims are assertions put forward for general acceptance. They're typically statements about a property of the system or some subsystem. Claims asserted as true without justification are assumptions, and claims supporting an argument are sub claims Arguments link evidence to a claim, which can be deterministic, probabilistic or qualitative. They consist of "statements indicating the general ways of arguing being applied in a particular case and implicitly relied on and whose trustworthiness is well established", together with validation of any scientific laws used. In an engineering context, arguments should be explicit. Evidence serves as the basis for justification of a claim. Sources of evidence can include the design, the development process, prior experience, testing (including statistical testing), or formal analysis.

In addition to the basic CAE concepts the framework consists of CAE Blocks that provide a restrictive set of common argument fragments and a mechanism for separating inductive and deductive aspects of the argumentation. These were identified by empirical analysis of actual safety cases, [5]. The Blocks are:

- Decomposition: Partition some aspect of the claim, or divide and conquer.
- Substitution: Refine a claim about an object into claim about an equivalent object.
- Evidence incorporation: Evidence supports the claim, with emphasis on direct support.
- Concretion: Some aspect of the claim is given a more precise definition.
- Calculation or proof: Some value of the claim can be computed or proved.

The framework also defines connection rules to restrict the topology of CAE graphical structures. The use of Blocks and associated narrative can capture challenge, doubts and rebuttal and to illustrate how confidence can be considered as an integral part of the justification.

The basic concepts of CAE are support by an international standard [1], IAEAE Guidance [3] and industry guidance [2]. To support CAE, a graphical notation can be used to describe the interrelationship of claims, arguments, and evidence (see [3] and [5]). In practice, top desirable claims such as "the system is adequately secure" are too vague or aren't directly supported or refuted by evidence. Therefore, it's necessary to create sub claim nodes until the final nodes of the assessment can be directly supported or refuted by evidence.

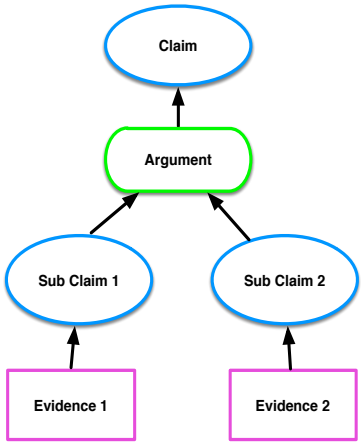

Figure 1: The CAE notation

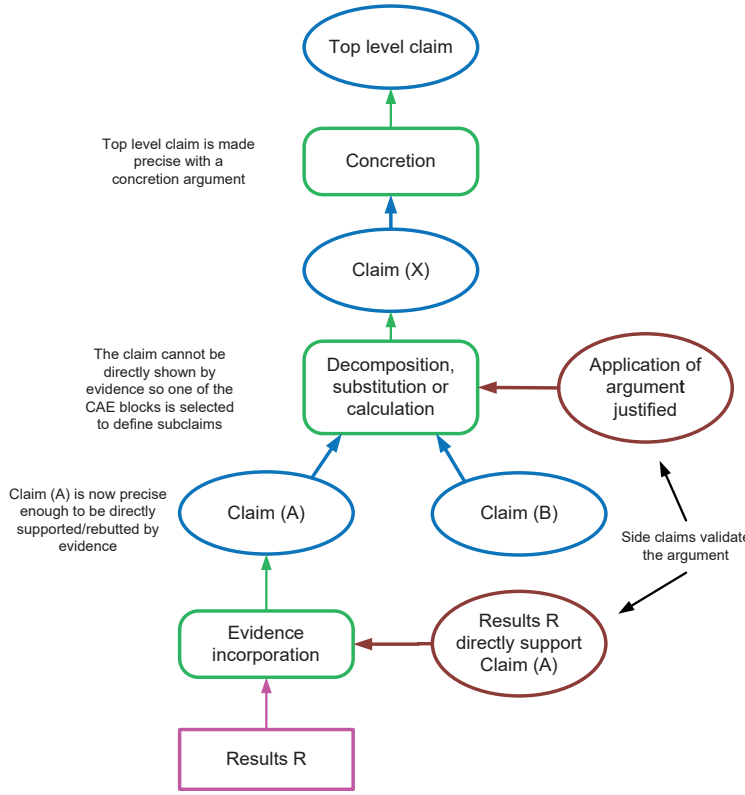

Figure 2: An example of CAE block use

Table 1: The assurance framework (ed: a callout box?)

The key advantage of a claim-based approach is that there is considerable flexibility in how the claims are demonstrated, since different types of arguments and evidence can be used as appropriate. Such a flexible approach is necessary when identifying gaps and challenges in uncharted territory, such as the assurance of ML-based systems. Indeed, CAE is commonly used in the safety-critical industry (e.g., defence, nuclear, medical, etc.) to assure a wide-range of systems and devices, and to support innovation in assurance.

We are developing a particular set of CAE structures that are generically applicable, and help identify how to develop trustworthy systems by explicitly considering evidence of sources of doubt, vulnerabilities, and mitigations addressing the behaviour of the system. In doing this we not only assure and determine challenges and gaps in behavioural properties but also self-identify gaps within the assurance framework itself. In the remainder of the article, we describe our systematic approach to identifying a range of gaps and challenges regarding ML-based systems and their assurance.

Identifying assurance challenges

The decision to trust an engineering system resides in engineering argumentation that addresses the evaluation and risk assessment of the system and the role of the different subsystems and components in achieving trustworthiness. Although previous abstractions, models, and relationships have been constructed in CAE for the assurance of traditional software systems, it is not clear if the said existing "blocks" are sufficient to provide compositional argumentation enabling trustworthiness in ML-based systems. For example, domain-specific abstractions and arguments may need to be developed in CAE to specifically target ML sub-components.

To develop a detailed understanding of such assurance challenges, we use CAE to develop an outline of an overall assurance case, proceeding from top-level claims, concerning an experimental autonomous vehicle and its social context, down to claims regarding the evaluation of subsystems, such as the ML model. The case study autonomous vehicle, as is typical with similar state-of-the-art vehicles, contains a heterogeneous mixture of commercial-off-the-shelf (COTS) components including image-recognition, LIDAR, etc. Apportioning the trustworthiness, dependability, and requirements of each of these components in order to consider the real-time and safety related nature of the system is challenging. In traditional safety critical engineering there would be diversity and defence in depth to reduce the trust needed in specific ML components yet we do not know whether this is practicable for ML based systems. Argumentation blocks may need to be further developed within CAE to determine how experimental data can allow for the comparison and assessment of diverse subsystems' contribution to defence in depth. This in turn can also inform future architectures of autonomous systems.

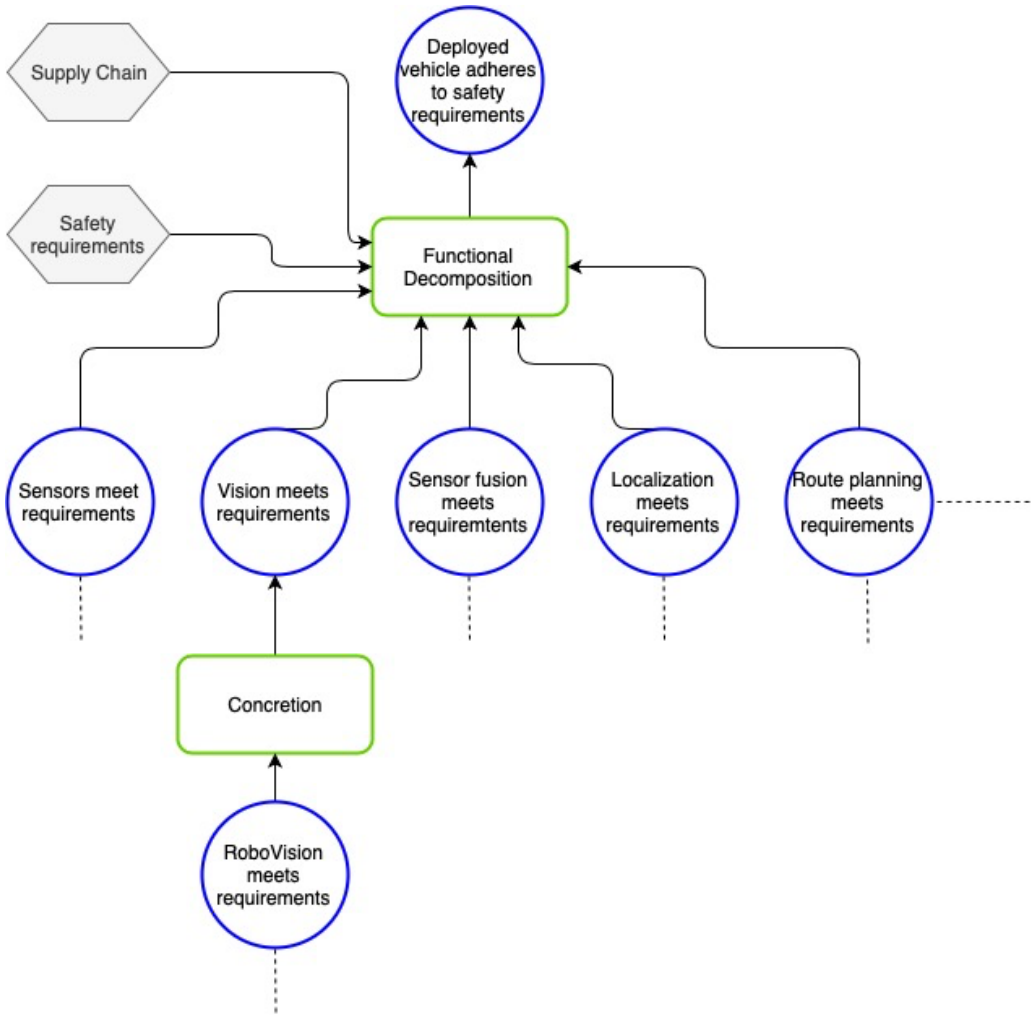

Figure 3: A high-level example of an assurance sub-case in CAE

Beyond the study of the applicability of CAE to assure ML-based systems, the lens of the assurance case is used to identify gaps and challenges regarding techniques and evidence aimed at justifying desired system behaviours. This is further informed by a review of literature, a case-study based assessment of the experimental vehicle, and an investigation of our industry partners' development processes in order to assess the current state of the vehicle and the short-to-medium term future vision of its use case (~2 years). To see how and whether security is addressed in the product lifecycle, we used the new UK Code of Practice PAS 11281 "Connected automotive ecosystems – Impact of security on safety" [9].

In the sections below, we discuss some of the gaps identified with regard to technical capabilities that may enable trust of system behaviours. We highlight three areas: requirements, security, V&V. There are also issues of ethics, advanced safety analysis techniques, defence in depth and diversity modelling that we do not address.

Gaps and challenges

Innovation and defining requirements and trust needed of the system:

There is a need to address the realities of the innovation lifecycle and to progressively develop requirements including those for trustworthiness and assurance. In this innovation approach, the vehicle is progressively developed from a platform to trial technologies to the final product. There is an assurance gap in that, when analysing how much the technologies need to be trusted, there needs to be an articulated vision of what they will be used for. If the vision of how something will be used is not articulated we cannot asses how much we need to trust it or what the risks are. This is particularly important for security and systemic risks, where the scale and nature of the deployment (e.g., as a key part of an urban transport system) will lead to more onerous requirements that need to be reflected in the earlier technology trials and evaluations. Alternatives more agile approaches would be to progressively identify these trust requirements as the innovation proceeds but this might lead to solutions that do not scale and in extreme could not be deployed.

We believe that the innovation lifecycle below is typical for many players in the industry and will be increasingly adopted as the ML components become more productised.

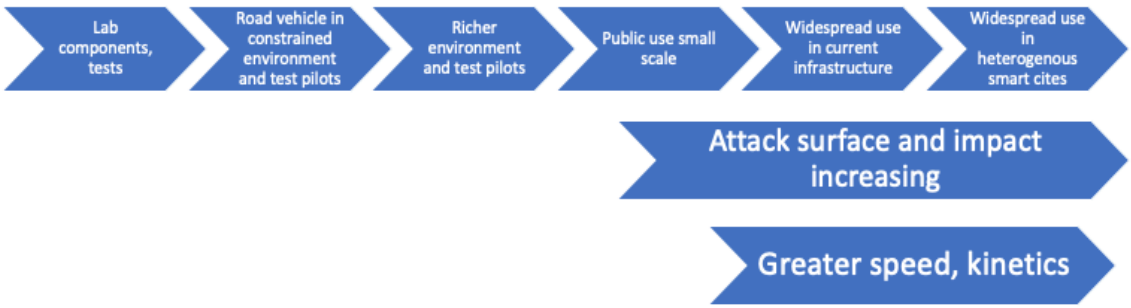

Figure 4: Typical stages of development from innovation to products

Security:

Security is a fundamental and integral attribute to the technical themes of the project: in the requirements, V&V, and assurance research. While the requirements of the new PAS 11281 Code of Practice may be met in a mature implementation of the vehicle being studied, on the whole, the security will be challenging for industry and there is a need to provide advice on partial and project specific implementation of the PAS that allow for maturity growth.

The security aspects need to be integrated into the entire lifecycle, as systems are not safe if they are not secure. This applies to the vehicle as a whole as well as to the ML subsystems: most ML systems have not been designed with a systematic attention to security [15]. The PAS clauses address

1. Security policy, organization and culture
2. Security-aware development process
3. Maintaining effective defences
4. Incident management

5. Secure and safe design
6. Contributing to a safe and secure world

And are equally applicable to the vehicle and its components.

As noted above, the deployment of autonomous technologies may follow an innovation lifecycle that first focuses on functionality and seeks to progressively add additional assurance and security. This will make the development of the assurance and safety cases and associated security and safety risk assessments particularly challenging. From our experience we recommend to:

1. Explicitly define the innovation cycle and assess the impact and feasibility of adding assurance and security.
2. Address the approach to security-informed safety at all stages of the innovation cycle. If safety, security and resilience requirements are largely undefined at the start of the innovation cycle, the feasibility of progressively identifying them during the innovation cycle should be assessed, together with the issues involved in evolving the architecture and increasing the assurance evidence.
3. Apply PAS 11281 to systematically identify the issues. Use a CAE assurance case framework and map PAS clauses to this to provide a systematic approach to applying the PAS.
4. Consider a partial and project specific implementation of the PAS to meet the innovation cycle.
5. Collect experience in developing a security-informed safety case and in integrating security issues into the safety analyses needed to implement the PAS.

Verification & Validation (V&V):

We use the assurance case in CAE top-down to identify the claims we wish to support and bottom up to evaluate the evidence that could be provided by them and hence systematically assess gaps, challenges and solutions: this is shown schematically in Figure 5. As part of this analysis we assessed state-of-the-art formal methods techniques for autonomous systems, and have observed that their maturity and applicability is lacking to sufficiently justify behavioural and vulnerability claims. Consider the issue of adversarial attacks and perturbations [10] [11], that has been particularly challenging with regard to the robustness of ML algorithms. Verification researchers have focused on the property of *pointwise robustness*, in which a classifier function *f'* is not robust at point *x* if there exists a point *y* within η such that the classification of *y* is not the same as the classification of *x*. That is, for some point *x* from the input, the classification label remains constant within the "neighbourhood" η of *x*, even when small value deltas (i.e., perturbations) are applied to *x*. A point *x* would not be robust if it is at a decision boundary and adding a perturbation would cause it to be classified it in the next class. Generally speaking, the idea is that a "neighbourhood" η should be reasonably classified as the given class.

However, proposed pointwise robustness verification methods [13][14][15] suffer from the same set of limitations:

- how to define meaningful regions η and manipulations
  - the neighbourhoods surrounding a point *x* that are currently used are arbitrary and conservative
- we cannot enumerate all *x* points near which the classifier should be approximately constant, that is, we cannot predict all future inputs

Furthermore, researchers have been unable to find compelling threat models that required perturbation indistinguishability [17], and it has been demonstrated that l_p, which defines the neighbourhood region η, is a poor proximity for measuring what humans actually see [18]. Finally, adversarial perturbations can be achieved by much simpler attacks that do not require ML algorithms (e.g., cover a stop sign). Thus, the extent to which these techniques can provide us with any level of confidence is not very high.

Other verification techniques [12][14] aim to verify more general behaviours regarding ML algorithms, instead of just pointwise robustness. Such techniques require functional specifications, written as constraints, to be fed into a specialized linear-programming solver to be verified against a piecewise linear constraint model of the ML algorithm. However, the generalization of these algorithms is challenging given the requirement of well-defined and bounded traditional system specifications, devoid of specifications regarding the behaviour of the ML algorithm itself. These techniques are thus applicable to well-specified deterministic ML algorithms, and cannot be applied to perception algorithms, which are notoriously difficult to specify, let alone, verify.

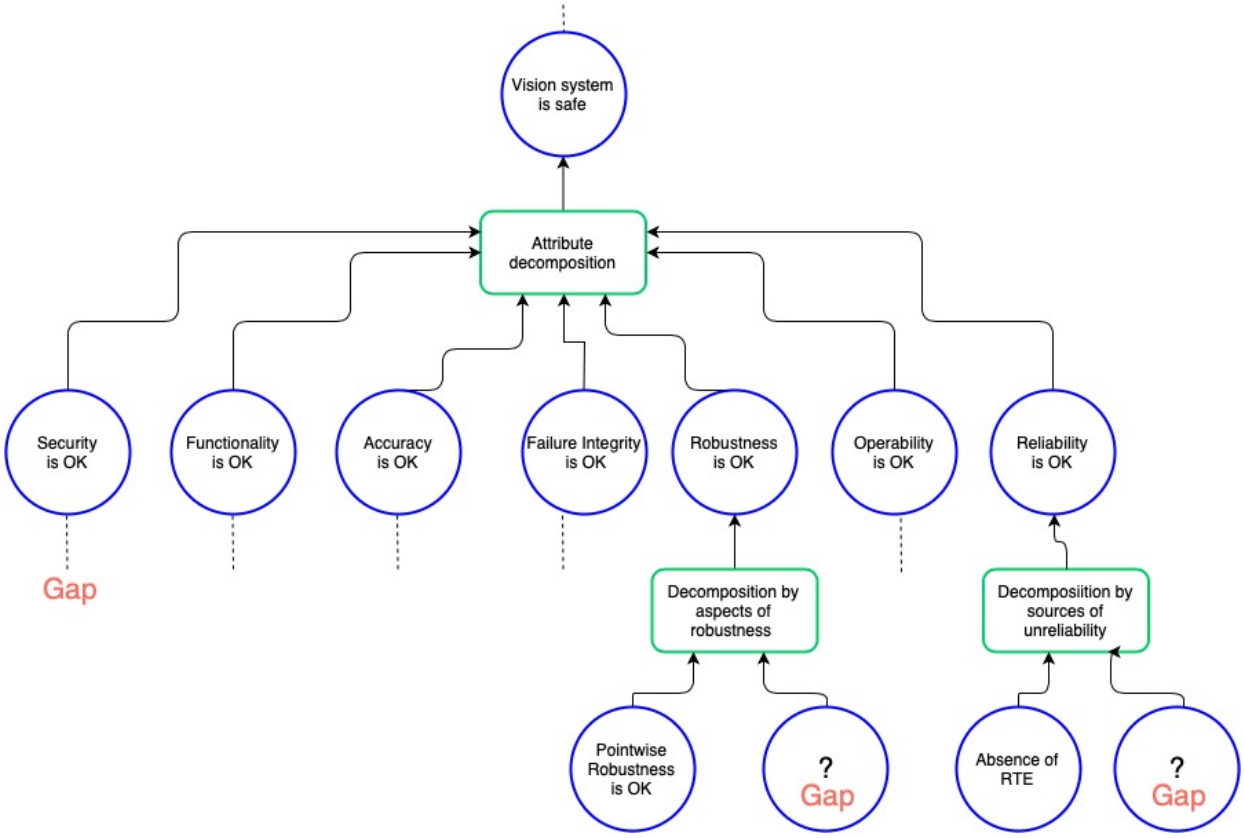

Figure 5: Using CAE to assess V&V gaps

Apart from the ML algorithm, the assurance of the non-ML supporting components of an autonomous system is challenging given that the use of COTS or open-source components leads to uncertain provenance. Errors within non-ML components can propagate and affect the functionality of the ML model [19]. It is therefore important to explore how traditional V&V methods, in particular, static analysis of C code, can provide assurance for the larger ML system, providing confidence beyond the component level. Below, we provide a preliminary list of results on analysing YOLO, a commonly used open source ML vision software, and a number of different run-time errors that were identified:

- A number of memory leaks, such as files opened and not closed, and temporarily allocated data not freed, leading to unpredictable behaviour, crashes, and corrupted data.
- A large number of calls to `free` where the validity of the returned data is not checked. This could lead to incorrect (but potentially plausible) weights being loaded to the network.

- Potential "divide by zeros" in the training code. This could lead to crashes during online training, if the system were to be used in such a way.
- Potential floating-point "divide by zeros", some of which were located in the network cost calculation function. As noted above, this could be an issue during online training.

Note that the above errors would only be applicable to languages such as C and C++. Not all these errors would be applicable to a language such Python, a popular language utilized in the implementation of numerous ML libraries and frameworks, as the semantics and implementation of the language itself does not enable overflow/underflow errors defined in [19]. However, Python is a dynamically typed language, bringing about a different set of program errors not exhibited by statically typed languages (e.g., type errors). Unfortunately, no static analysis techniques or tools exist to allow for the analysis of Python code. Furthermore, it is unclear how potential faults arising from dynamic languages could affect the functionality of an ML model itself. Indeed, this is a large gap within the formal methods field that needs to be addressed immediately, given the deployment of autonomous vehicles utilizing Python.

Discussion and conclusion

We believe there is need for disruptive innovation in the assurance of autonomous and machine learning based systems. We have provided a summary of the outcome focussed, claims arguments evidence based framework we are evolving to address these systems and have used it to identify specific gaps and challenges as well as discussing some solutions. We have demonstrated the feasibility of deploying the best of existing work (e.g. advanced static analysis techniques) and identified the need for new approaches.

Overall there is a need for a stronger evidence base of the dependability of machine learning components and of the autonomous systems as a whole. There is common good in sharing data on algorithms, training data, benchmarks, reliability growth, and incidents in sufficient detail for independent analysis and research. We hope to play our part in this by sharing curated data and generic assurance cases and by providing in the public domain the more detailed report this article is based on.

And lastly, if we can achieve our goal of disruptive assurance this can have a positive impact on innovation in other industries and technologies, not just ML based ones

Acknowledgements

The article discusses work undertaken within the Towards Identifying and closing Gaps in Assurance of autonomous Road vehicleS (TIGARS) project. The project is a collaboration between Adelard, Witz, City University of London, the University of Nagoya, and Kanagawa University. We acknowledge the support of the Assuring Autonomy International Programme and the UK Department for Transport.

References

[1] ISO/IEC 15026-2:2011. Systems and software engineering — Systems and software assurance, Part 2: Assurance case (2011)

[2] Bishop, P.G., Bloomfield, R.E.: A Methodology for Safety Case Development. In: Redmill, F., Anderson, T. (eds.) Industrial Perspectives of Safety-critical Systems: Proceedings of the Sixth Safety-Critical Systems Symposium, Birmingham 1998, pp. 194–203. Springer, London (1998)

[3] Dependability Assessment of Software for Safety Instrumentation and Control Systems at Nuclear Power Plants" (NP-T-3.27), https://www-pub.iaea.org/books/IAEABooks/12232/Dependability-Assessment-of-Software-for-Safety-Instrumentation-and-Control-Systems-at-Nuclear-Power-Plants

[4] Rushby, J. The Interpretation and Evaluation of Assurance Cases, Technical Report SRI-CSL-15-01, July 2105

[5] Bloomfield, R. and Netkachova, K. "Building blocks for assurance cases", S 2014 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW),3-6 Nov. 2014 doi: 10.1109/ISSREW.2014.72

[6] Kalra, Nidhi and Susan Paddock. Driving to Safety: How Many Miles of Driving Would It Take to Demonstrate Autonomous Vehicle Reliability?. Santa Monica, CA: RAND Corporation, 2016. https://www.rand.org/pubs/research_reports/RR1478.html.

[7] Koopman, P., and M. Wagner. 2016. "Challenges in autonomous vehicle testing and validation". SAE International Journal of Transportation Safety 4 (2016-01-0128): 15-24.

[8] Robin Bloomfield, Peter Bishop, Eoin Butler, Robert Stroud, Security-informed safety – supporting stakeholders with codes of practice, Cybertrust column, IEEE Computer, August 2018

[9] BSI PAS 11281 - Connected automotive ecosystems

[10] Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., & Fergus, R. Intriguing properties of neural networks. arXiv preprint arXiv:1312.6199, 2013.

[11] Goodfellow, I., Shlens, J., Szegedy, C. "Explaining and harnessing adversarial examples," in International Conference on Learning Representations. Computational and Biological Learning Society, 2015.

[12] Pulina, L., & Tacchella, A. An abstraction-refinement approach to verification of artificial neural networks. In International Conference on Computer Aided Verification (pp. 243-257). Springer Berlin Heidelberg, July 2010.

[13] Huang, X., Kwiatkowska, M., Wang, S., & Wu, M. Safety Verification of Deep Neural Networks. arXiv preprint arXiv:1610.06940, 2016.

[14] Katz, G., Barrett, C., Dill, D., Julian, K., & Kochenderfer, M. Reluplex: An Efficient SMT Solver for Verifying Deep Neural Networks. arXiv preprint arXiv:1702.01135, 2017.

[15] Papernot, N., McDaniel, P., Sinha, A., & Wellman, M. Towards the Science of Security and Privacy in Machine Learning. arXiv preprint arXiv:1611.03814, 2016.

[16] Ruan, W., Huang, X., & Kwiatkowska, M.Z. Reachability Analysis of Deep Neural Networks with Provable Guarantees. IJCAI, 2018.

[17] Gilmer, J., Adams, R., Goodfellow, I., Andersen, D., Dahl, G. Motivating the Rules of the Game for Adversarial Example Research. Gilmer, J., Adams, R.P., Goodfellow, I.J., Andersen, D., & Dahl, G.E. (2018). Motivating the Rules of the Game for Adversarial Example Research. CoRR, abs/1807.06732.

[18] Wang, Z. and Bovik, A. C. "Mean squared error: Love it or leave it? A new look at signal fidelity measures". In: IEEE Signal Processing Magazine 26.1 (2009), pp. 98– 117.

[19] Robustness Testing of Autonomy Software, C Hutchison et al. IEEE/ACM 40th International Conference on Software Engineering: Software Engineering in Practice Track (ICSE-SEIP), May/June 2018.