Transfer Assurance for Machine Learning in Autonomous Systems

Chiara Picardi[1,*], Richard Hawkins[1], Colin Paterson[1] and Ibrahim Habli[1]

[1]University of York, York, UK

Abstract

This paper introduces the concept of *transfer assurance* for Machine Learning (ML) components used as part of an autonomous system (AS). In previous work we developed the first approach for assuring the safety of ML components such that a compelling safety case can be created for their safe deployment. During operation it may be necessary to update an ML component by re-training the model using new or updated development data. If model re-training is required post-deployment, the safety case that was created for the ML component may no longer be valid, since a new model has been created that can no longer be assured to meet its safety requirements. In particular, the nature of machine learnt components means that one may not be able to predict how even small changes in the development data may affect the model and its performance. As a result, current practice would require that a full assurance assessment is undertaken for the re-learned model, and that a new safety case is created. Given the desirability of updating ML components during operation, we see it as imperative that the assurance process become more proportionate to the size of the change that is made to the model, whilst ensuring that assurance can still be demonstrated. Retraining ML components is known to be a costly and complex process and as such techniques such as transfer learning have been developed which aim to reduce this burden through incremental development. Approaches such as transfer learning provide an inspiration for how the challenge of efficiently assuring updated models could be addressed through understanding which aspects of a model may have been affected by changes to the development data. We refer to this as transfer assurance, where parts of the assurance case for an ML component can remain fixed whilst other parts are re-assessed.

Keywords

Assurance, Safety, Machine Learning, Transfer Learning, Deep Learning

1. Introduction

The use of ML models promises to revolutionise a number of societally-significant, safety-critical domains including healthcare, transport and defence [1, 2, 3]. Whilst such systems may be reported to exceed human performance [4, 5], their adoption is dependent on establishing justified confidence in the safety of systems not only at development time but also once deployed in complex open-world environments.

In our previous work we developed a methodology, called AMLAS (Assurance of Machine Learning for use in Autonomous Systems) that systematically integrates safety assurance into the development of ML components in order to generate a safety case to demonstrate the ML component is safe to use in a particular autonomous system (AS) application [6]. Through evaluative studies [7, 8, 9, 10] it has been shown that through following AMLAS it is possible to produce an assurance case which is valid for deployment.

The assurance case that is created for the ML component is based upon predictions and assumptions made during development about the system and the environment in which the ML component will be used. These predictions and assumptions may turn out during operation of the system not to be true, or may become untrue as things change in unexpected ways. When this happens the safety case created during development may no longer be valid. It is necessary therefore to identify when such changes occur and what the impact of those changes may be on the ML component and therefore on the safety of the system.

During operation it may be necessary to update an ML component by re-training the model using new or updated development data. There are a number of reasons why it may be desirable to update a learned model after deployment. For example, as more data becomes available during operation of the AS, it may be possible to use that data to improve the performance of the model. It may also be the case that during operation the operational inputs to the ML model are observed to diverge from the development data used in the model learning process.

Such shifts in the distribution of the operational data with respect to the development data may require that the model is re-trained with data sets that more closely reflect the current operational data in order to ensure that the model continues to perform as required in the operational context [11]. A further consideration is that the

SafeAI, February 13–14, 2023, Washington, DC

*Corresponding author.

chiara.picardi@york.ac.uk (C. Picardi); richard.hawkins@york.ac.uk (R. Hawkins); colin.paterson@york.ac.uk (C. Paterson); ibrahim.habli@york.ac.uk (I. Habli)

safety case for the ML component is based upon predictions and assumptions about the environment in which the ML component will be used. Due to the complexity and dynamicity of the operating environment of the autonomous system, there is the possibility that the predictions and assumptions may be incorrect, or become invalid as the environment evolves in unexpected ways [12]. Again this may require model re-training to ensure the assurance case remains valid.

In each of these cases, if the model is re-trained post-deployment, the safety case that was created for the ML component may no longer be valid, since a new model has been created that can no longer be assured to meet its safety requirements. In particular, the nature of machine learnt components means that one may not be able to predict how even small changes in the development data may affect the model and its performance. As a result, current practice would require that a full assurance assessment is undertaken for the re-learned model, and that a new safety case is created. This would involve applying the whole of the AMLAS process again following any change to the ML component. Given the desirability of updating ML components during operation, we see it as imperative that the assurance process become more proportionate to the size of the change that is made, whilst ensuring that assurance can still be demonstrated.

Retraining ML models is known to be a costly and complex process and as such techniques have been developed which aim to reduce this burden through incremental development. Transfer learning is one such technique that enables large parts of a convolutional neural network (CNN) to remain fixed during re-learning, whilst some of the layers are learned using the updated development data. Approaches such as transfer learning provide an inspiration for how the challenge of efficiently assuring updated models could be addressed through understanding which aspects of a model may have been affected by changes to the development data. We refer to this as *transfer assurance*, where parts of the assurance case for an ML component can remain fixed whilst other parts are re-assessed. This paper conceptualises transfer assurance and looks at how a transfer assurance approach can be used to re-assure the ML model and update the safety case for the ML model without having to apply the whole AMLAS process.

The rest of the paper is structured as follows. In Section 2 we discuss related research. Section 3 introduces the transfer assurance process that we propose. Section 4 discusses some of the key open challenges related to transfer assurance that we aim to address. Section 5 presents our conclusions.

2. Related Work

2.1. Model Updating

The need to update models at run-time is well understood, however model updating presents a number of challenges [13, 14] including:

1. deciding when a model is no longer valid
2. suitable mechanisms for retraining

Model retraining can be scheduled to occur periodically when the environment in which the system operates can be assumed to be stable for a fixed period of time. However when the environment is less predictable, mechanisms need to be employed which are able to detect the potential for a deterioration in system performance through the monitoring of data to identify distribution shifts [15]. In order to detect shifts in data distributions we can monitor the performance of the model, when the ground truth is known, or the input/output distributions in order to detect any significant changes [16] which may invalidate assumptions made at design time.

Once the need to retrain has been identified we need to establish how to undertake such retraining. Two approaches are proposed in the literature:

1. **offline** using batches of data [13], or
2. **online**, continuously updating the model in operation [17].

In both cases the dataset used for retraining contains both new and historical data in order to avoid catastrophic forgetting [18]. In this paper we consider only offline retraining methods.

2.2. AMLAS

AMLAS is a methodology for the assurance of machine learning in autonomous systems [6]. AMLAS provides a set of defined processes that describe activities that should be undertaken as well as the artefacts that are used by, or generated from those activities. AMLAS also provide a set of safety argument patterns that describe how these artefacts are used to create a compelling safety case for the ML component. AMLAS is split into six lifecycle stages as shown in Figure 1. The ML assurance activities run in parallel to the ML development activities; the assurance activities defined in AMLAS should be an integral part of developing an ML component. Feedback and iteration between the stages of AMLAS is also important. It is expected that it might be necessary to revisit stages of AMLAS multiple times. For example, the verification stage might identify a need to revisit data management and then learn a new model, and so on.

Figure 1 also shows the input to the AMLAS process from the system safety requirements. Safety considerations for the ML component are only meaningful when

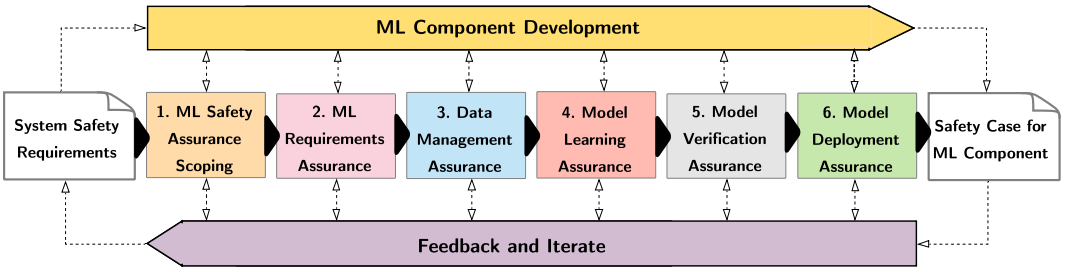

Figure 1: AMLAS assurance process in context [6]

scoped within the wider system and operational context. The system safety requirements provide that crucial link between AMLAS and the system safety process. Having followed the AMLAS process the outputs from each stage can be integrated together to form an overall safety case for the ML component. [9] provides an example how AMLAS has been used to create a safety case for a neural network used to detect wildfires. AMLAS can be used as mean for compliance with standards as part of emerging regulatory frameworks [19] [20].

2.3. Assurance of ML at Run-time

There has been some previous work that has investigated run-time assurance of ML systems ([21], [22]), however these typically focus on particular applications such as reinforcement learning for self-driving cars [23] or ML for robotic systems [24]. The approach proposed in [25] uses monitors to determine if safety critical requirements are violated by ML components at run-time and provides feedback to improve the models.

None of this work considers how an assurance case is impacted by changes at run-time or how the validity of the assurance case is maintained. Other work [12] has highlighted the need to dynamically update safety cases in operation, but has not considered how this could be done for ML components.

3. Transfer Assurance Process

In this section we discuss in detail the proposed 3 stage transfer assurance process shown in Figure 2.

The first stage concerns the initial development of the ML component for deployment into the operational system. The two activities result in the creation of an ML component along with its assurance case and a set of appropriate monitors to be deployed to the autonomous system.

The second and the third phases deal with analysing and responding to changes during operation of the autonomous system. These phases will be executed multiple times during the lifetime of the system as and when the monitor issues a response trigger. In the following subsections we explain in detail each of the proposed stages.

3.1. Initial Development

The Initial Development stage contains two activities generating two artefacts as well as the ML component and a set of monitors.

The first of the activities creates the ML component and the associated assurance case using the AMLAS process [6]. The resulting assurance case report is used in the second activity in order to create a component monitor. The monitor is created by considering the evidence, assumption, context and justification elements present in the assurance case to identify those environmental and system conditions that, if they were to change, would invalidate the safety argument. The data to be monitored may be concerned with environmental changes outside the autonomous system, changes in physical or software components which constitute the autonomous system platform or in the behaviour of one or more ML components enacting autonomous behaviours.

Thus a monitoring strategy is created which considers how the nature of potential changes in operational data will impact the assurance case to undermine safety assurances. The strategy then defines what data is to be monitored, the nature of the sensors to be used, and the analytical basis upon which a response trigger should be generated.

As an example let us consider an object classification component which is deemed acceptable in the assurance safety case under the assumption that the risk to be controlled is a function of the impact caused by misclassification and likelihood of encounter. Verification of the

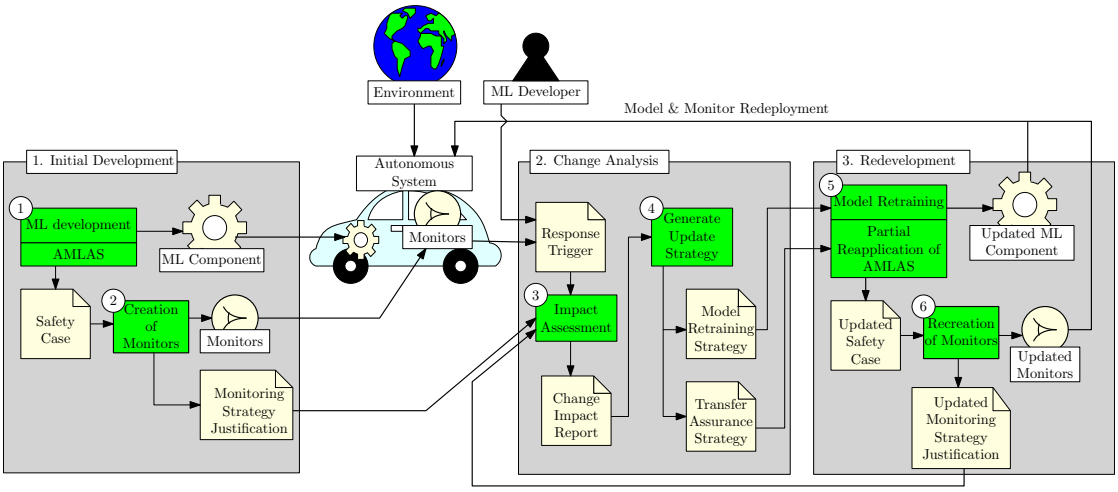

Figure 2: Three Stage Transfer Assurance Process with activities shown in green and artefacts in yellow.

component shows that an accuracy of 0.87 and assumptions of the operating domain determine a likelihood of encounter rate of one per 10 hours of operation. A monitor for likelihood may then be created which counts the number of class predictions such that a trigger is raised should we identify that there has been a distribution shift in the probability of class encounter.

We can also consider the example of an ML model for speech recognition trained on adult voices. The assurance argument will have a context element defining the training data as adult voices. This can then be used to identify the requirement for a monitor to recognise if the input to the model is not an adult voice, as it cannot be assured that the model will meet its safety requirements when exposed to non-adult voices as input.

There are two main problems with creating monitors for ML systems during operation. Firstly it is difficult to detect distribution shifts in complex inputs such as images. Secondly it is often difficult to monitor the correctness of the output of the model due to unavailability of ground truth. We can overcome these problems using sensor fusion, contextual information and established beliefs. In the rest of the subsection we illustrates some methods using examples.

Typically autonomous systems make use of multiple sensors during normal operation and we may be able to use these to either directly or indirectly infer properties of interest. Specifically sensor agreement and comparison can be used to check the validity of a sensor readings at run-time. For example we may consider a drone which utilises a 6 channel multi-spectral camera for agricultural monitoring [26]. Whilst each channel may have independent analysis pipelines, using ML components with different functional requirements, the output from each sensor is correlated since it refers to the same physical objects in space. From development activities we will have extracted expectations for the distributions of component outputs for each of the 6 channels in operation. For example when Channels 1, 3 and 4 predict outcome A then with probability 0.98 Channel 2 will agree. Through monitoring the combinatorial behaviour characteristics of sensors we may be able to identify distribution shifts in the data which compromise safety at run-time.

Whilst components are typically quoted with point values for expected performance (e.g. accuracy, precision) perturbations encountered at run-time can significantly impact ML performance [27]. Understanding the link between real world phenomena and component performance allow for the creation of sensors which monitor such phenomena and allow for the validation of operating assumptions at run-time. Consider for example an image classifier utilised to identify road signs in an autonomous vehicle. Experimental results at development time may provide a performance profile with respect to the amount of fog, or mist, in the air [28]. A simple moisture monitor deployed at run-time may be employed to update the anticipated accuracy of the sensor and to say when the sensor falls outside it's safe operating mode, hence triggering a redevelopment of the component and the associated assurance case.

The output of an ML component can also be monitored using contextual information combined with knowledge about the physical world in order to estimate ground truth. For example, in a self-driving car an object detection component identifies pedestrians with an associated bounding box which is used to estimate the position and size of the pedestrian. This estimate is updated as each frame of the video is presented to the component and

the movement of the box reflects behaviours in the real-world which must be consistent with physical laws. For example the pedestrian may not 'disappear' or suddenly 'relocate' to impractical areas of the scene. When such changes are detected this is indicative of a failure of the sensing pipeline. In this way data consistency may be used to estimate false positives, area of union or mean average precision at run-time.

Another way to estimate the ground truth is combining sensor predictions with contextual meaningful information. As an example let's consider a self-driving car with an assumption in the assurance case asserting that the frequency of false positive on speed limit signals shall be less than a certain threshold. On a motorway in the UK a vehicle is unlikely to encounter a 30 mph speed sign. Using a GPS unit on the car confirms the position and ,combining this with the contextual information, we can determine that if a 30 miles per hour speed limit is detected this represents a false positive.

In order to assess safety during operations usually the precursor events or leading indicators (i.e. events likely to happen before an accident) are monitored [29]. In other words monitoring the output of the system detecting precursor event can be a good indication that the ML component is not safe as required. As an example we could monitor the frequency of emergency stops contextually knowing that these should not happen frequently or in self-driving cars we could easily monitor the disengagement rate knowing that if the car hand-over the control to the human driver too often then something is not working as it should. When a precursor event happens frequently, the development team should perform a casual analysis to understand where the problem could be in the pipeline.

3.2. Change Analysis

The change analysis stage seeks to evaluate the impact of any change that activates a response trigger. The response trigger may be flagged by a monitor, or be initiated by an ML developer who has identified a desire to change the model (for example to improve model performance during operation). Once the change is triggered, the impact of the change is analysed to determine whether updates to the model, or to the safety case, or to both are required. Where an update is required, a strategy for doing so in an effective manner is determined.

Changes will generate a response trigger if they have been determined to be potentially important with respect to the validity of the safety case. When a response trigger is activated during the operation of the autonomous system, it is necessary to assess the nature and extent of the impact the change has on the safety case. It should be noted that the impact assessment activity is not expected to be an automated process. The assessment will require an understanding of the reason that the monitor was created (informed by the monitoring strategy justification), as well as of the safety case itself. The assessment therefore relies upon human expertise and judgement. The change impact report should therefore be used to document how the impact of the change was determined.

Where the change has an impact on the safety case it is necessary to determine an update strategy that will ensure a valid safety case can be maintained. It may be that the impact on the safety case can be mitigated without requiring an update to the model itself. For example, system level mitigation may be possible, such as by introducing additional operating restrictions on the autonomous system. These restrictions would serve to ensure that, for example the system was not exposed to the situations that triggered the change response. In such cases the scope of the safety case can be updated to reflect the change in operation.

Where it is determined that model retraining is required in order to mitigate the impact of the change on the safety case, an appropriate update strategy must be determined. It is desirable to create an update strategy that limits the amount of costly and time consuming assurance rework that is required (such as a need to re-apply the whole of the AMLAS process). In the transfer assurance approach we propose that the model retraining strategy and the assurance strategy should be closely linked, such that the model retraining strategy enables a reduction in the assurance effort required to re-establish a valid safety case.

3.3. Redevelopment

In the redevelopment stage we update the ML model and its associated safety case in line with the recommendations from the previous stage.

While the majority of ML learning is stateless [15], i.e. the model is re-learnt from scratch each time, such approaches result in "catastrophic forgetting" where all previously learnt knowledge is discarded. Given the high cost of building models this is undesirable and where knowledge remains valid across model iterations it is desirable to retain knowledge encoded in the model. In an attempt to achieve this techniques, such as transfer learning [30], have been developed which allow for appropriate features of a model to be retained whilst also allowing for new features to be learnt. One benefit of transfer learning is to reduce the amount of data required for model development as well as development costs.

Undertaking the AMLAS process from scratch each cycle is also to be avoided if possible and, where we can control the scope of retraining, we may also reduce the assurance effort. For example, the identification of a new subclass may lead us to gather a small set of data and relabel some existing data. These activities will require us to

justify the accuracy of data labelling and the collection of appropriate data, however undertaking new assurance activities associated with historic data or "frozen" portions of the model may be unnecessary.

A transfer assurance approach would allows us to reconsider only a subset of the AMLAS activities originally undertaken in stage 1 of the proposed process. The updated safety case resulting from the partial reapplication of AMLAS will, however, introduce new evidence and may modify assumptions, contexts and justification elements. These changes may necessitate the updating of existing monitors as well as the creation of new monitors for deployment. In addition each modified monitor will have an associated monitoring strategy justification document which will inform subsequent impact assessment activities.

Once the new model and associated monitors have been deployed we return to operation and await a new response trigger to undertake stages 2 and 3 once more.

4. Addressing Open Challenges

In order to deliver transfer assurance we must address a number of open challenges which will be the focus of future work.

In stage 1 of the proposed approach we suggest a number of techniques which may be employed to monitor environments for distributional shifts which invalidate assurance claims. In open environments however such changes are combinatorial in nature and identifying impact factors in high dimensional space remains challenging. Being able to identify small regions of the input space will however limit the need for new data to be generated for training and potentially allow for larger regions of the machine learnt component to be retained between redevelopment cycles. In addition an understanding of the factor interactions in decision space will improve the specification of monitoring strategies,reduce the number of response triggers and potentially reduce the time taken for developing assurance cases.

In stage 2 of the process we proposed the development of update strategies through an analysis of the trigger response and monitoring strategy justification document. There exists no guidance at present concerning how trade offs and design decisions should be derived. For example we may become aware that the performance of a component with respect to a subclass is insufficient but it is unclear at present as to whether one is better to a) gather new data b) label sub classes or c) restrict the scope of assurance. Each option has costs and benefits and guidance on how to decide the correct course of action in a contextually relevant manner is needed.

In stage 3 we recognise that continual learning and transfer learning strategies are in sight for many real world applications and that research in this area is well underway. We do not, at present however, have a strategy for partial assurance of incrementally learnt models. We intend to address this by building on the work of AMLAS to construct processes and safety patterns to guide the assurance of such models.

In future work we intend to address these challenges with application to a range of real world problems where we will consider: the needs of different operating contexts including autonomous driving and healthcare; a range of model forms including neural networks and reinforcement learning policies; and different modes of change including evolutionary drift and step changes in the environment.

5. Conclusions

In this paper we proposed a process by which we may update, simultaneously, an ML model integrated into an autonomous system and its assurance case as defined prior deployment. Such an approach is necessary due to the evolutionary nature of the open contexts into which autonomous systems are being deployed and a desire to ensure safety through life.

We have presented a number of examples of how the approach may be practically achieved and highlighted open challenges and opportunities for future work. We believe that transfer assurance promises to allow for reduced development and assurance costs, more responsive deployment cycles and ultimately safer autonomous systems in real-world contexts.

6. Acknowledgements

This work was supported by the Engineering and Physical Sciences Research Council through the RAILS project (EP/W011344/1) and the Assuring Autonomy International Programme, a partnership between Lloyd's Register Foundation and the University of York.

References

[1] J. De Fauw, et al., Clinically applicable deep learning for diagnosis and referral in retinal disease, Nature medicine 24 (2018) 1342–1350.

[2] S. Grigorescu, et al., A survey of deep learning techniques for autonomous driving, Journal of Field Robotics 37 (2020) 362–386.

[3] S. A. Azizan, I. A. Aziz, Terrorism detection based on sentiment analysis using machine learning, Journal of Engineering and Applied Sciences 12 (2017) 691–698.

[4] M. Nagendran, et al., Artificial intelligence versus clinicians: systematic review of design, reporting standards, and claims of deep learning studies, bmj 368 (2020).

[5] C.-J. Farrell, Identifying mislabelled samples: machine learning models exceed human performance, Annals of Clinical Biochemistry 58 (2021) 650–652.

[6] R. Hawkins, et al., Guidance on the assurance of machine learning in autonomous systems (AMLAS), arXiv preprint arXiv:2102.01564 (2021).

[7] S. Laher, et al., Review of the AMLAS methodology for application in healthcare, arXiv preprint arXiv:2209.00421 (2022).

[8] J. Hamnert, D. Hägglund, How safe is machine vision?: An evaluation of the AMLAS process in a machine vision environment, 2022.

[9] R. Hawkins, et al., Creating a safety assurance case for an ML satellite-based wildfire detection and alert system, arXiv preprint arXiv:2211.04530 (2022).

[10] P. Festor, et al., Assuring the safety of ai-based clinical decision support systems: a case study of the ai clinician for sepsis treatment, BMJ Health & Care Informatics 29 (2022) e100549.

[11] S. G. Finlayson, et al., The clinician and dataset shift in artificial intelligence, The New England journal of medicine 385 (2021) 283.

[12] E. Denney, G. Pai, I. Habli, Dynamic safety cases for through-life safety assurance, in: 2015 IEEE/ACM 37th IEEE Int. Conf. on Software Engineering, volume 2, IEEE, 2015, pp. 587–590.

[13] D. Klabjan, X. Zhu, Neural network retraining for model serving, arXiv preprint arXiv:2004.14203 (2020).

[14] A. Paleyes, R.-G. Urma, N. D. Lawrence, Challenges in deploying machine learning: a survey of case studies, ACM Computing Surveys (CSUR) (2020).

[15] C. Huyen, Designing Machine Learning Systems, " O'Reilly Media, Inc.", 2022.

[16] J. Klaise, et al., Monitoring and explainability of models in production, arXiv preprint arXiv:2007.06299 (2020).

[17] I. Prapas, et al., Continuous training and deployment of deep learning models, Datenbank-Spektrum 21 (2021) 203–212.

[18] R. M. French, Catastrophic forgetting in connectionist networks, Trends in cognitive sciences 3 (1999) 128–135.

[19] Artificial intelligence/machine learning (ai/ml)-based software as a medical device (samd) action plan, https://www.fda.gov/media/145022/download, 2021. Accessed: 2022-12-12.

[20] EASA concept paper: First usable guidance for level 1 machine learning applications, https://www.easa.europa.eu/en/downloads/134357/en, 2021. Accessed: 2022-12-12.

[21] D. Cofer, et al., Run-time assurance for learning-enabled systems, in: NASA Formal Methods Symposium, Springer, 2020, pp. 361–368.

[22] S. Chen, et al., Runtime safety assurance for learning-enabled control of autonomous driving vehicles, in: 2022 Int. Conf. on Robotics and Automation (ICRA), IEEE, 2022, pp. 8978–8984.

[23] P. Musau, et al., On using real-time reachability for the safety assurance of machine learning controllers, in: 2022 IEEE Int. Conf. on Assured Autonomy (ICAA), IEEE, 2022, pp. 1–10.

[24] A. Desai, et al., Soter: a runtime assurance framework for programming safe robotics systems, in: 2019 49th Annual IEEE/IFIP Int. Conf. on Dependable Systems and Networks (DSN), IEEE, 2019, pp. 138–150.

[25] P. Mallozzi, Combining machine-learning with invariants assurance techniques for autonomous systems, in: 2017 IEEE/ACM 39th Int. Conf. on Software Engineering Companion (ICSE-C), IEEE, 2017, pp. 485–486.

[26] J. Franke, G. Menz, Multi-temporal wheat disease detection by multi-spectral remote sensing, Precision Agriculture 8 (2007) 161–172.

[27] J. Mohapatra, et al., Towards verifying robustness of neural networks against a family of semantic perturbations, in: Proceedings of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition, 2020, pp. 244–252.

[28] C. Paterson, et al., Deepcert: Verification of contextually relevant robustness for neural network image classifiers, in: Int. Conf. on Computer Safety, Reliability, and Security, Springer, 2021, pp. 3–17.

[29] N. Leveson, A systems approach to risk management through leading safety indicators, Reliability engineering & system safety 136 (2015) 17–34.

[30] C. Tan, et al., A survey on deep transfer learning, in: Int. Conf. on artificial neural networks, Springer, 2018, pp. 270–279.