



Deposited via The University of York.

White Rose Research Online URL for this paper:

<https://eprints.whiterose.ac.uk/id/eprint/228098/>

Version: Accepted Version

Article:

Paterson, Colin, Hawkins, Richard David, Picardi, Chiara et al. (2025) Safety assurance of Machine Learning for autonomous systems. Reliability Engineering and System Safety. 111311. ISSN: 0951-8320

<https://doi.org/10.1016/j.ress.2025.111311>

Reuse

This article is distributed under the terms of the Creative Commons Attribution (CC BY) licence. This licence allows you to distribute, remix, tweak, and build upon the work, even commercially, as long as you credit the authors for the original work. More information and the full terms of the licence here:

<https://creativecommons.org/licenses/>

Takedown

If you consider content in White Rose Research Online to be in breach of UK law, please notify us by emailing eprints@whiterose.ac.uk including the URL of the record and the reason for the withdrawal request.

Safety Assurance of Machine Learning for Autonomous Systems

Colin Paterson^a, Richard Hawkins^a, Chiara Picardi^a, Yan Jia^a, Radu Calinescu^a, Ibrahim Habli^a

^a*Department of Computer Science, University of York, Heslington, York, YO10 5GH, United Kingdom*

Abstract

Machine Learning (ML) components are increasingly incorporated into systems, with different degrees of autonomy, where model performance is reported as meeting, or exceeding, the capabilities of human experts. This promises to transform products and services, in diverse domains such as healthcare, transport and manufacturing, to better serve underrepresented groups, reduce costs, and increase delivery effectiveness, especially where expert resources are scarce. The greatest potential for transformative impact lies in the development of autonomous systems for safety-critical applications where their acceptance, and subsequent deployment, is reliant on establishing justified confidence in system safety. Creating a compelling safety case for ML is challenging however, particularly since the ML development lifecycle is significantly different to that employed for traditional software systems. Typically ML development involves replacing detailed software specifications with representative data sets from which models of behaviour is learnt. Indeed, ML's strength lies in tackling problems which are challenging for traditional software development practices. This shift in development practices introduces challenges to established assurance processes which are crucial to developing the compelling safety case required for regulation and societal acceptance. In this paper we introduce the first methodology for the Assurance of Machine Learning for use in Autonomous Systems (AMLAS). The AMLAS process describes how to systematically and attractively integrate safety assurance into the development of ML components and how to generate the evidence base for explicitly justifying the acceptable safety of these components when integrated into autonomous system applications. We describe the use of AMLAS by considering how a safety case may be constructed for an object detector for use in the perception pipeline of an autonomous driving application. We further discuss how AMLAS has been applied in several domains including healthcare, automotive and aerospace as well as supporting policy and industry guidance for defence, healthcare and automotive.

Keywords: Machine Learning, Safety, Assurance, Safety Case, Autonomy

1. Introduction

The use of machine learning (ML) in everyday services and products has grown rapidly, with recent breakthroughs in deep learning supporting the development of ML models capable of performance levels that match or exceed those of human experts in tasks ranging from medical diagnosis [1, 2] to financial trading [3] and marine life monitoring [4]. As a result, these models are increasingly considered for a variety of roles in systems where their failure could compromise safety. In particular, ML components are well suited to performing key tasks required in autonomous systems, and difficult to achieve using traditional software. Examples of such tasks include perception in autonomous vehicles [5, 6, 7], process optimisation and supply chain management in manufacturing [8], maintenance of critical infrastructure [9, 10, 11], hazard identification in autonomous shipping [12, 13] and reliability prediction in nuclear power plants [14]. Unfortunately, these uses of ML have not been without failures [15, 16], leading to accidents and, in some cases, to loss of life. Successful adoption of ML for such safety-related tasks therefore hinges on our ability to assure that the ML will be safe to use prior to its deployment into operation. This requires understanding of the ways in which the ML may contribute to safety risk for the system [17, 18], and how that contribution is acceptably managed.

The established methodologies for the safety assurance of traditional software components of safety-related systems are not applicable to ML components, which are developed using a completely different lifecycle [19] than standard software. This lifecycle includes multiple stages (e.g., data management, and model training) and activities

(e.g., data collection, preprocessing and augmentation, and ML model and hyperparameters selection) that are not encountered in traditional software development. For that reason, the safety assurance of machine learning requires new, dedicated methodologies that consider these stages and activities explicitly. This paper introduces such a method.

For novel technologies such as ML, where there is no established practice for safety and considerable variation in design approaches, it is crucial that, as well as identifying the safety processes and techniques required to manage safety risk effectively, system developers are able to explain and demonstrate to a range of stakeholders how sufficient safety has been achieved. Safety cases provide a way of explicitly demonstrating, through the creation of a structured argument supported by a compelling body of evidence, why a system is safe to operate. The safety case argument can be used to explain the sufficiency of the adopted ML safety assurance processes. When software technology underwent a similarly fast adoption in safety-related applications, safety cases were similarly identified as the best approach [20, 21, 22]. Safety cases have been used for the assurance of systems from application domains as diverse as aerospace [23], rail transportation [24], defence [25] and nuclear power generation [26]. This success has led to safety cases being enshrined in standards [27, 28], and to research on methodologies for their rigorous development. This has included the development of safety argument patterns as a way to guide the structure of safety cases.

Our paper introduces the first methodology that focuses on the Assurance of Machine Learning for use in Autonomous Systems (AMLAS). Underpinned by a systematic six-stage process, AMLAS employs a suite of new safety argument patterns to produce a complete safety case for a given ML component of an autonomous system. AMLAS details the activities undertaken, and the artefacts (e.g., requirements, data sets and assurance evidence) required by, and generated at, each stage of a typical ML development lifecycle [19]¹. Crucially, AMLAS explains explicitly how these artefacts are used to instantiate the associated safety argument patterns for the ML component, and to generate its safety case. This safety case can then be used to support safe deployment decisions for ML in autonomous systems.

The first two stages of the AMLAS process define the scope under which safety assurance for the ML component is needed, and provides assurance for the safety requirements of this component, respectively. The next three stages concentrate on the assurance of the data sets used to train, validate and test the ML component, on the actual training of the ML model, and on its verification. Finally, the AMLAS process ends with a stage that provides safety assurance for the deployment of ML component into the operational autonomous system.

The rest of the paper is structured as follows. In Section 2, we provide the necessary background on safety cases and safety argument patterns. Section 3 introduces a use case from the autonomous driving domain which is used to illustrate the application of the AMLAS methodology, detailed in Section 4. In Sections 5 and 6, we present results obtained from the application of AMLAS to other contexts and the lessons learnt from these applications. Finally, Section 7 presents our conclusions and proposes directions for future work.

2. Background

This section provides background information on safety cases and safety argument patterns which are used throughout the paper. We also present an introduction to supervised machine learning, which is used extensively in a range of autonomous systems and which underpins the worked example used in the paper to illustrate our approach.

2.1. Safety Cases

A safety case provides a set of evidence along with a clear, comprehensive and defensible argument in support of a given claim about the safety of a system within a particular operating context [29]. Since their introduction, safety cases have become a requirement in numerous safety standards and regulations for industries including nuclear energy [30], medical devices [31] and air-traffic control [32]. Given the diversity of application domains, each safety case must be specific to the system and processes for which the argument is to be constructed. Indeed the development of a safety case is a complex process itself comprising activities performed across all stages of the development life cycle. In practice any safety case is presented with explicit assumptions and contextual scoping such that the arguments only hold when these contexts also hold.

¹We note that within this work we are primarily concerned with traditional ML components, e.g. Convolutional neural networks for classification tasks, rather than Generative AI models such as Large Language models.

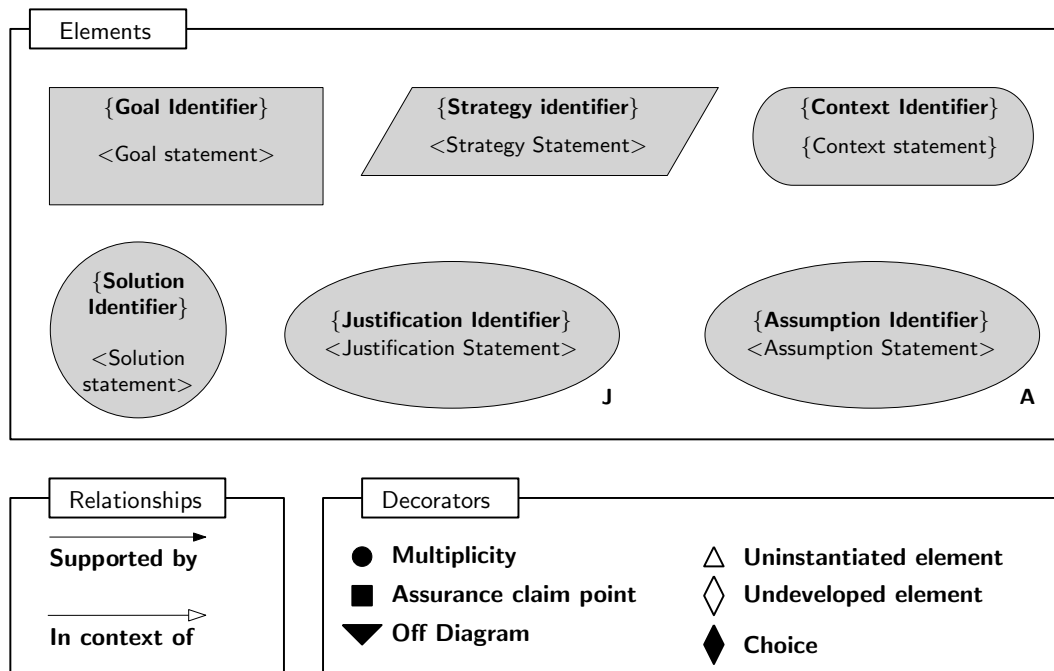


Figure 1: Core concepts from the GSN standard [33]

Goal Structuring Notation (GSN) is a standardised graphical notation [33] which has found use widely in industry for the presentation of assurance cases [34]. The main concepts used to construct a safety case using GSN are shown in Figure 1. A claim made as part of a safety case is shown in GSN using the goal element, which may be presented with relationships to assumptions and contexts within which the claim may be said to hold. The goal structure then decomposes this goal into sub-goals until we reach a point where a solution may be presented. A solution consists of evidence that the goal is met. The decomposition of goals may be undertaken using a strategy which is presented with an appropriate justification, and throughout the argument the context within which concepts exist and the assumptions underpinning the claims are made explicit. For a detailed description of GSN we refer the interested reader to the GSN standard [33].

2.2. Safety Argument Patterns

The generation of a safety argument requires considerable effort and the reuse of argument approaches from one safety case to another is to be expected. Reuse of safety arguments can reduce the effort required for argument generation, and ensure the dissemination of good practice within safety case development. The reuse of arguments may, however, lead to potentially dangerous outcomes if the arguments are used inappropriately. To mitigate this, explicit safety argument patterns can be used [35] that enable the essence of the required argument and evidence to be captured, whilst abstracting away from the details of a particular application.

Safety argument patterns can be documented using GSN to capture an argument structure that represents good practice and which is believed to be applicable to a more general set of applications. Within a pattern, certain aspects of GSN elements are left uninstantiated. The uninstantiated aspects are indicated by the presence of curly brackets in the text, and an uninstantiated element decorator. When using a safety argument pattern in the creation of a safety case, uninstantiated aspects of the argument pattern must be instantiated using information that is specific to the system or application under consideration. The undeveloped element decorator may also be used within the pattern to indicate that further work will be needed to determine appropriate support for a claim for the particular system.

2.3. Supervised Machine Learning for Autonomous Systems

The primary focus of AMLAS is the safety assurance of ML components obtained through *off-line supervised learning* and in this section we introduce the concepts which are central to supervised learning. Supervised machine

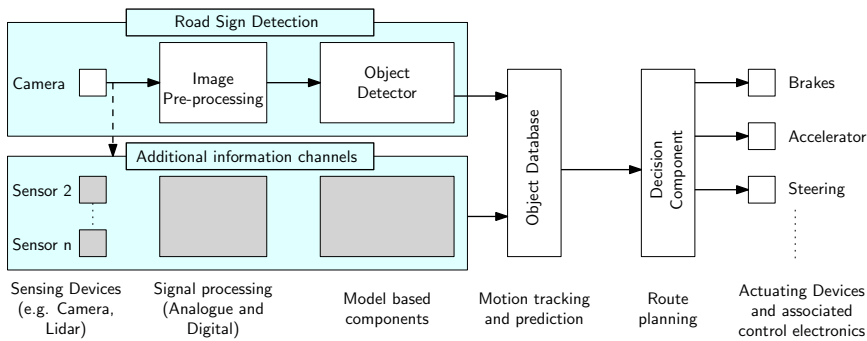


Figure 2: Abstract representation of an autonomous driving platform in which multiple information channels are derived from sensor inputs to control actuating functions.

learning techniques are at the forefront of the emergence of autonomous systems, underpinning classification and object recognition tasks for use in domains as diverse as autonomous driving [36] and medical diagnosis [37]. At its core supervised learning is the task of learning a mapping between a set of training samples x_i and a pre-defined set of labels y_i .

If we define a training set T to be a set of n input-output pairs

$$T = (x_1, y_1), (x_2, y_2), \dots (x_n, y_n)$$

where we assume that each output y_i was generated by an unknown function $y = f(x)$ applied to the input, then the aim of supervised learning is to learn a function h which approximates f .

For an image classification task, the sample x_i is typically a vector where each element in the vector is the pixel value of a colour channel in the image. For a standard HD video camera a single frame would, therefore, present a sample with over 6 million values ($1920 \times 1080 \times 3 = 6,229,800$). In practice images of this size are problematic and so pre-processing pipelines are employed to segment or scale the sample before learning. For classification problems the label associated with the image is then typically encoded as an ordinal value where labelling is typically a manual processes. Whilst classification is a common problem for supervised learning this technique may also be used to learn regions of an image or, on numeric data, statistical features.

The central aim of machine learning is generalisation, that is learning a function that not only works well on the training data but also works well on that previously unseen data [38]. This problem drives our choice of model, our choice of training data and the learning techniques employed. For many problems, finding h is formulated as an optimization problem in which an objective, or loss, function is created to measure how far the estimated function is from the true function. Learning then involves minimising loss through the adjustment of model parameters. Whilst reducing the problem in this way allows for solutions to be found, assuring their safety is challenging and throughout the remainder of this paper we will demonstrate how AMLAS allows for such assurance arguments to be constructed.

3. Example Use Case

In order to introduce AMLAS we make use of a worked example derived from the world of autonomous driving. Specifically we consider the task of identifying road traffic signs. In autonomous vehicles a perception pipeline is typically constructed such that machine learning components transform data from sensors to produce semantically meaningful information for use in a decision framework [39]. An abstract representation of such an autonomous driving platform is shown in Figure 2 and illustrates the use of multiple, parallel, information processing channels to control the behaviour of the vehicle.

We assume that the primary information channel for the road sign detection pipeline processes data from a video camera. The raw data feed from a video camera may be processed to normalise light levels, remove sensor noise etc. before the image is resized for use in the object detection algorithm. The object detection algorithm may be considered (conceptually) as having two functions as shown in Figure 3. In the first stage the video frame is analysed

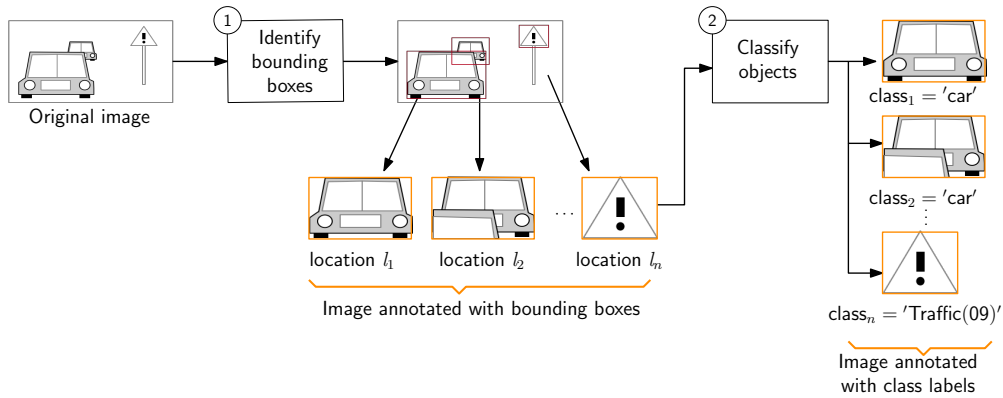


Figure 3: Conceptual workflow of a two stage object detection algorithm in which the first stage identifies a set of bounding boxes for each object in the scene and the second stage allocates labels to each identified object.

and a vector $\vec{l}_i = (x_1, x_2, y_1, y_2)$ is created for each of the n objects detected where the x and y co-ordinates define the corners of the box. In the second stage, the contents of each bounding box is passed to an object classifier which allocates a label to the object.

The information produced by the object detector may then be used, in conjunction with data from other information channels, to update an object database which is responsible for not only tracking objects of interest in the operational space, but also estimating the location and trajectory of those objects. The capacity of the object database is finite and therefore merging and discarding objects is necessary. Finally the information about the current state of the world is passed to a decision framework which derives appropriate actions which are enacted by a set of actuating devices and their control electronics [40].

Object detection is a supervised learning problem and, as such, the creation of the ML component requires that we obtain a training dataset which has been labeled with both bounding boxes and class labels. Since this is a vision system deployed into an open world context the range of possible inputs, including environmental perturbations, is very large and providing a complete set of all possible inputs is not possible. A machine learnt object detector may provide inaccurate, or incorrect, results in a number of ways. For example, a failure in the first stage may lead to a stop sign being missed such that a vehicle enters a junction at speed. Alternatively a sign may be correctly identified but in the wrong location. Consider a vehicle on a motorway approaching a junction where a sign applies only to vehicles leaving the motorway. In such a case stopping abruptly on the main carriageway may lead to an accident. Failures in the classification stage of the detector may also lead to unsafe behaviours. For example, confusing 30mph and 60 mph speed signs, or misclassifying a stop sign as an give way sign. The likelihood and impact of each type of machine learnt component failure depends on the context in which the system is deployed as well as the operating platform into which the component is deployed.

4. Methodology

The AMLAS process, shown in Figure 4, starts by considering the system safety requirements. These requirements are obtained through the application of system-level safety engineering [41], specifically the safety requirements generated from hazard identification and risk analysis. Whilst broader discussion of safety engineering is beyond the scope of this work we provide guidance on how safety assurance may be systematically integrated into the development of autonomous systems for use in complex environments in other work [42]. These requirements are passed to the first stage of the process in which we define the bounds within which the safety of the ML component can be demonstrated. In the next stage we translate these component agnostic safety requirements into ML specific requirements and demonstrate their equivalence. In stage 3 we consider the use of data as an encoding of component specifications and define the requirements for data used to train, validate and test the ML model in order to achieve the defined ML safety requirements. Next stage 4 considers the model creation process by which the data is combined with a learning strategy to create models which will meet the safety requirements when deployed in the target system.

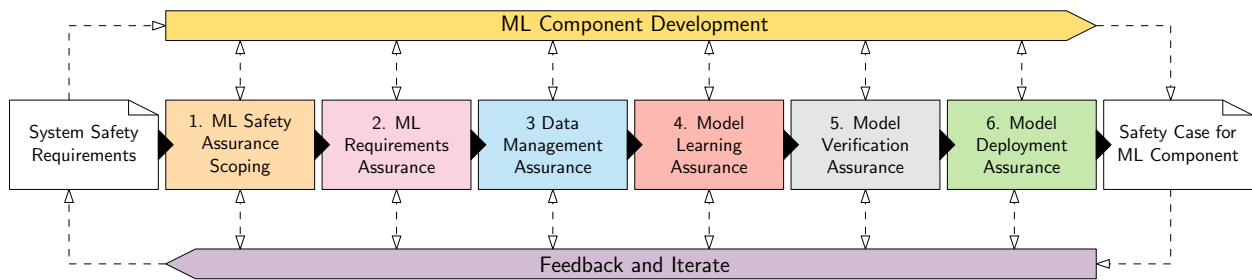


Figure 4: Overview of the six-stage AMLAS process. The colour of each stage is reused in each pattern and process associated with the stage.

The candidate model is then verified in stage 5 to ensure the model will meet the allocated safety requirements when exposed to inputs not present in the training data. Finally in stage 6 we determine whether the ML component will continue to meet the specified system level safety requirements when deployed into the operational system.

The AMLAS assurance activities are intended to be undertaken in parallel with development processes. In this way evidence gathered from both development and assurance activities are combined to generate an explicit safety case for each ML component by using the safety argument patterns provided for each stage of AMLAS. AMLAS is, by its nature, an iterative process so the output from any single stage may lead to a reconsideration of the previous stages. For example the results of the verification stage may highlight conditions under which ML safety requirements are violated. By analysing these conditions, we can identify whether we need to modify our robustness requirements (stage 2) and then gather additional data (stage 3) to improve the model.

4.1. ML Safety Assurance Scoping

AMLAS provides assurance for the safety of ML components. This is different from making claims about reliability or performance of ML. Safety is inherently context-sensitive. Safety claims can only ever be made with respect to the context of the system in which ML will be used, the role it plays within that system, the environment in which the system operates and the safety hazards that are present. This additional information is therefore essential to understanding the safety of the ML component, and is the reason that this ML safety assurance scoping stage is so important as part of the AMLAS process.

The aim of this stage is then to define the scope under which we are able to demonstrate the safety of the resulting ML component.

Figure 5 shows the scoping process consisting of 2 activities and 7 artefacts consumed, and generated, by the AMLAS activities. The first activity defines the scope of the safety assurance for the ML component by analysing the component’s function, the systems it will be used in, the operating environment, and the associated system safety requirements. Based on this analysis, the safety requirements that apply to the component can be identified and allocated. At this stage, the requirement is independent of any ML technology or metric. Instead, it reflects a need for the component to perform safely within the system and operating environment regardless of the technology later deployed. The component description is functional in nature, stating the intended operational characteristics of any component deployed in this role.

The second activity then uses the GSN pattern, shown in Figure 6, and the artefacts from the previous activity to instantiate an ML assurance scoping argument.

Central to the pattern is claim **G1.1**, which states that the component will meet the safety requirements defined in activity 1. For the worked example one may instantiate the placeholder text in **G1.1** with “The object detection component”. The pattern states that this goal exists within contexts defined in **C1.1** - **C1.4**. Each of these contexts are then instantiated with the relevant artefact. For example, once instantiated, context **C1.2** may say “System and architecture report 2.04” i.e. the document which forms artefact **[C]**. As discussed above, correctly establishing this contextual information is important to the validity of the safety case, and ensuring the information is complete and correct for the application domain. For the complex open-world contexts in which many safety-critical autonomous systems operate, defining a complete description of the operational environment is particularly challenging. It is essential therefore that this contextual information is derived from discussions between system safety engineers and ML developers to avoid assumptions which would invalidate the safety case. It may therefore be appropriate to

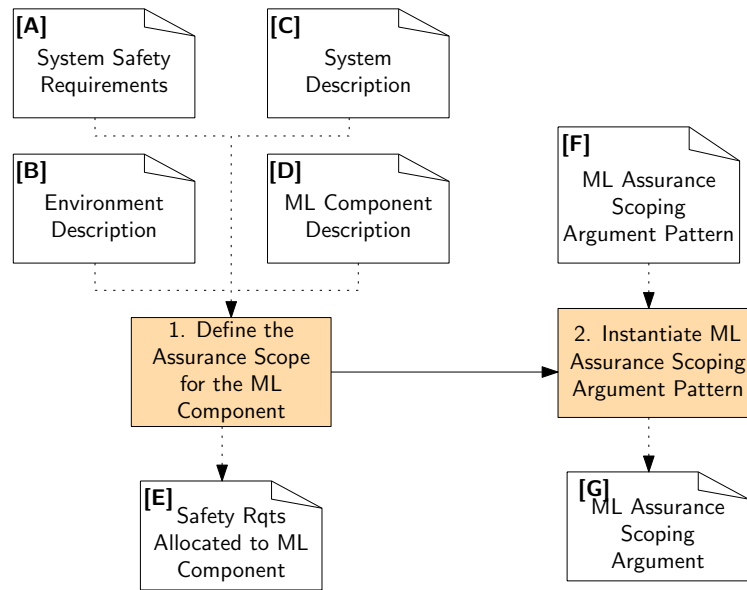


Figure 5: The AMLAS ML Assurance Scoping Process showing 2 process activities and their associated artefacts. The letters in square brackets ([A] - [G]) are artefact identifiers which are mapped to the assurance patterns.

provide a confidence argument to justify the sufficiency of each of these contextual artefacts within an ML context. To maintain clarity of the argument pattern however we choose not to represent these confidence arguments explicitly. In this pattern the assumptions that underpin the claims made are stated explicitly in **A1.1**. In this case the assumption is that the system safety process has correctly identified the system safety requirements allocated to the ML component.

Finally the pattern states the strategy, **S1.1**, by which **G1.1** will be supported. The strategy is to split the argument into two parts. Firstly the development of the ML component is considered, beginning with the development of ML safety requirements as outlined in activity 2. Secondly the deployment of the ML component will be addressed by stage 6 of the AMLAS process. The way in which the strategy is enacted is described by two patterns which are linked at the bottom of Figure 6 and described in Sections 4.2 and 4.6.

The artefacts that have been generated from the Stage 1 activities are used to provide this context in the safety argument. The outcome of this AMLAS stage is then an instantiation of the argument pattern in Figure 6 where the uninstantiated aspects, in curled braces, will be replaced with details of the ML component under consideration to create the first part of the safety case. In the next section we consider Stage 2 in which the AMLAS process produces the next part of the safety case considering the ML safety requirements.

ML Safety Assurance Scoping for the Use Case

For our worked example we shall consider the requirement for the vehicle to obey all traffic signs. The safety requirements will vary for different types of traffic sign, based on the required response of the vehicle. For example speed limit signs may require the vehicle to change speed, warning signs may require a change of lane, and so on. For our example we will consider stop signs, where the vehicle must recognise their presence in sufficient time to safely come to a halt. As the ML component that we are considering is responsible for object detection, we must identify the safety requirement on the object detection component relating to stop signs.

Determining the safety requirements on the component involves vehicle safety analysis activities which are largely outside of the scope of this paper, however it is important to consider the way in which the safety requirements are derived. If the vehicle fails to detect the presence of a stop sign this would clearly be hazardous since the vehicle may fail to stop as required which could result in a collision. It is firstly necessary therefore to consider, from a safety perspective, how many missed stop sign would be considered acceptable. This may be determined through, for example, comparison to the performance of a careful and competent human driver. This enables us to prescribe, for instance, a permissible number of stop sign misses per 1000 miles of driving. For the object detector component, we are interested in the required probability of failing to detect any single stop sign. We must therefore also consider the

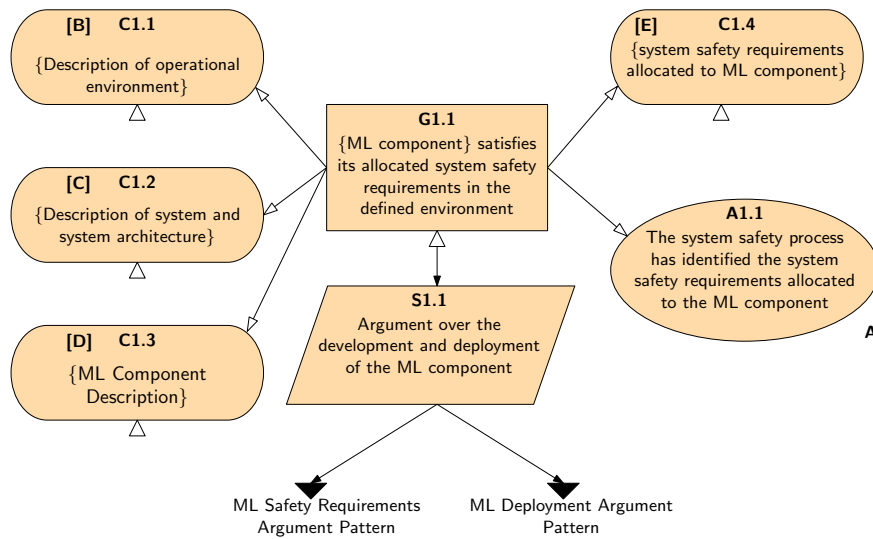


Figure 6: The ML Assurance Scoping Argument Pattern where the letters in square brackets ([B] - [E] in the top left of the GSN elements are present to aid the reader and indicate links to artefacts in the assurance process given in Figure 5.

number of stop signs that the vehicle is expected to encounter per 1000 miles of driving.

As described in Figure 2, the input to the object detector component is a series of image frames obtained from a video camera. This means that for each stop sign that is present, the object detector will be presented with multiple frames. The detection component then outputs a detection result for each frame which updates the belief that a stop sign is present in the operating environment. The decision component will, therefore, have multiple opportunities to determine the presence of each stop sign. Therefore, based upon the frame rate of the input images, and the decision framework implemented by the decision component, it becomes possible to specify how many frames of detection are required, per stop sign, in order for the vehicle to respond safely.

An additional consideration is how close the vehicle will be to the stop sign before it is detected. To be safe, this must allow the vehicle enough time to stop comfortably without excessive braking. Determining this distance will require assumptions to be made regarding the maximum speed that the vehicle may be travelling, and the worst-case braking performance of the vehicle in adverse road conditions.

Based on an analysis of these factors for the vehicle we derive the following safety requirement for the object detection component:

“SR1 - The object detector component shall correctly detect any stop sign present on the planned path of the vehicle in its correct location in 95% of frames where a stop sign exists within 80 metres of the vehicle.”

False detection of stop signs is also a safety concern since a vehicle stopping inadvertently increases the potential for rear-end collisions. In addition any unpredictable behaviour from a vehicle can lead to unsafe behaviour in other vehicles, such as unnecessary and sudden overtaking. A false detection may arise if the perception component indicates a sign that is not actually present, or mis-classifies another object as a stop sign (such as a tree or an advertising board).

As for missed detections, specifying safety requirements related to false detection of stop signs will require an analysis based upon assumptions regarding the vehicle systems and the operating environment. This enables us to specify the following additional safety requirement for the object detection component:

“SR2 - The probability of the object detector identifying a stop sign on the planned route, where no such object exists, shall be less than 1% in each frame.”

The position of a sign within a road network can be important in understanding to which vehicles the sign is applicable. In dense and complex road configurations, identifying a sign in an incorrect position could lead to an unnecessary response, and therefore potentially hazardous behaviour. Determining the safety requirement relevant to this requires an understanding of anticipated road geographies and sign orientations using the environment description [B]. For autonomous driving systems such as this, the details of the operating environment are often captured through

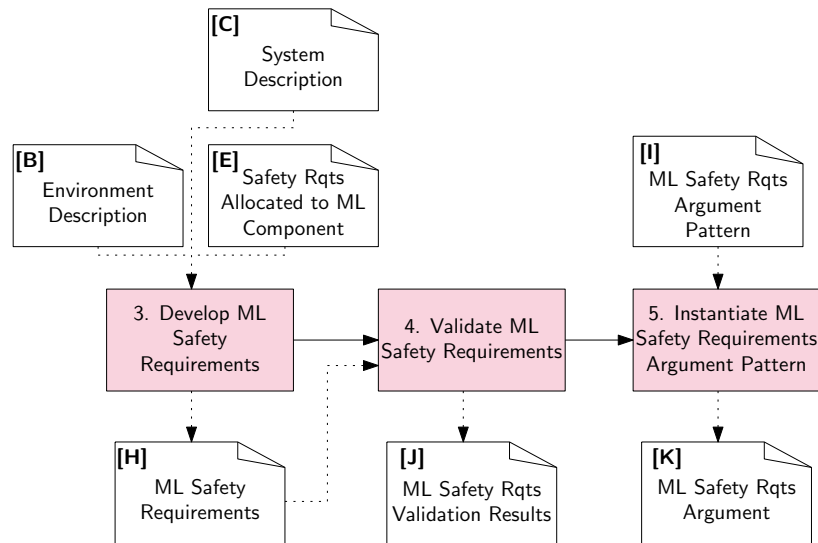


Figure 7: The AMLAS ML Requirements Assurance Process consisting of 3 process activities and their associated artefacts.

the specification of an operational design domain (ODD) [43] [44]. The derivation of appropriate behaviours for the system can therefore only be undertaken where the description of the ODD is complete. Assuring the completeness of the ODD is a subject of ongoing research [45] This enables us to specify one more safety requirement for the object detection component:

“SR3 - The object detector component shall detect stop signs within 1.2 metres of their true position.”

These requirements must be shown to be satisfied within the defined context including the operational environment (C1.1) and for the system and its architecture (C1.2).

4.2. ML Requirements Assurance

Having completed Stage 1 and identified the relevant component level safety requirements, we must now translate these into requirements which are suitable for use in the construction, and verification, of a machine learnt model. One of the primary focuses of this stage is to justify the sufficiency with which the derived ML safety requirements capture the intent of the component level safety requirements. We must be able to demonstrate that the ML requirements have properly accounted for the previously identified component level safety requirements, architecture and environment description.

This stage of the process, shown in Figure 7, consists of 3 activities and 7 artefacts consumed and generated in the enactment of this stage.

The first activity in the stage, activity 3, consumes the safety requirements for the component as generated in the previous stage, as well as the system and environment descriptions, and derives requirements in a form suitable for machine learning. This requires translating complex real world concepts and cognitive decisions into a format and level of detail that is amenable to ML implementations and verification [46]. The biggest challenge we face in implementing autonomous systems using ML is in replacing, or augmenting, human decision making, which has general intelligence, with an ML component with specific capabilities. This requires information which is normally implicit to be made explicit as part of the ML requirements. In AMLAS we expect as a minimum that the set of ML safety requirements will include requirements on both the performance and robustness of the ML model.

Performance of machine learnt models is typically defined as a mathematical quantity which represents a threshold which must be met or exceeded by the model in operation. As part of AMLAS, it is important that this threshold provides a representation of safe operation as allocated to the model. In practice it is rarely possible to define a performance threshold for safety using a single value as a threshold, so multiple metrics must be considered when defining the threshold. Additionally it is often necessary to require a trade off between the performance metrics. For

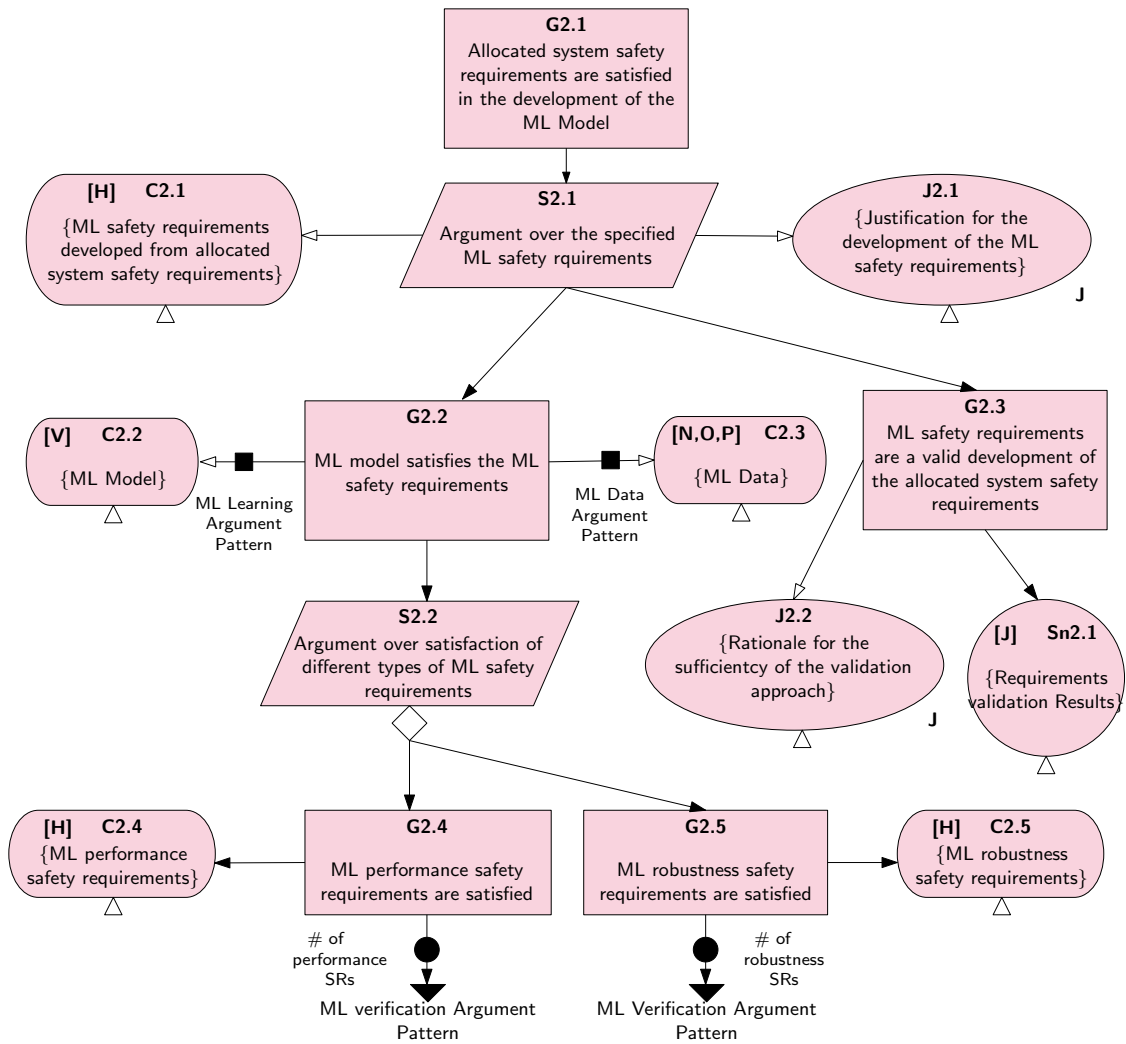


Figure 8: Argument Pattern for ML Safety Requirements

example in a healthcare application for the detection of medical conditions in patient scans it is necessary to define an acceptable balance between false negatives and false positives [47] which reflects sufficiently safe performance.

Model robustness considers the model's ability to perform well when the inputs encountered during operation are different but similar to those present in the training data. To argue the robustness of a model with respect to the system safety requirements requires us to consider how any, even small, changes in the operational input space may impact the performance of the model. Therefore robustness requirements will encapsulate features of the operational space that have been identified as having an effect of the model output. This will include defining dimensions of variability for each of the relevant features of the operational space for example, environmental features [48], sensor noise [19] and the variation of humans in the loop [49]. The range of variability will be specified by domain experts to cover the situations likely to be encountered at run-time within the operational context (as captured in the operational environment description).

We note that whilst the development of an ML model necessitates the consideration of numerous functional and non-functional requirements, e.g. the energy requirements of a computationally complex ML algorithm at run-time, in AMLAS we limit our consideration to those requirements which impact the operational safety of the system.

Activity 4 takes the previously defined ML safety requirements and validates them with respect to the allocated system safety requirements in the defined system and environmental contexts. There are two commonly used ap-

Table 1: Operating Domain Features and Dimensions of variability for ML component robustness

Feature	Variability
Weather : Rain	{none, low, med, high}
Weather : Fog	{none, low, med, high}
Lighting : Glare Angle	{none, 0, 30, 60}
Lighting : Ambient	{ normal, bright, dark }
Environment : Obscuration	{none, 10%, 50%}
Environment : Damage	{ none, minor, severe}

proaches for validating ML safety requirements. Firstly using reviews by domain experts to ensure that the specified ML safety requirements for the component will deliver the intended system level safety requirements. Secondly through simulation of a constructed system in which the ML safety requirements are obeyed and outcomes are observed for a set of operational scenarios.

The final activity of this stage constructs the ML safety requirements argument using the argument pattern given in Figure 8 and the artefacts from the previous activities. We note that the ML safety requirements developed in activity 3 provide the context for the strategy employed to assure our top level goal. The justification **J2.1** should explicitly explain the issues involved in translating complex real world concepts and cognitive decisions into formats amenable to ML implementation. For our motivating example this may include mapping distances the geometry of the real world to pixel regions in the image for example. Our strategy here is then to derive to sub goals where **G2.3** considers the validity question while **G2.2** focuses on the ML safety requirement in the context of the model and data each using by an assurance claim point (ACP) which we provide as argument patterns developed later in the process. The satisfaction of the safety requirements is decomposed based on the different types of requirement and will include claims concerning the performance and robustness of the machine learnt component.

ML Requirements Assurance for the Use Case

Let us consider the three allocated safety requirements from the first stage of AMLAS (*SR1, SR2, SR3*) which consider the components ability to identify and locate objects in the input image. Machine Learnt object detectors, such as YOLO [50], typically return a set of bounding boxes and associated class labels for each video frame. Mean average precision (mAP) [51] is commonly used to assess component performance and incorporates measures of recall, precision and localisation accuracy. In this way the single metric combines a weighted assessment of the three safety requirements specified in stage 1. We note that localisation in the object detector is not measured as a distance in metres in the real world, or a projection into pixel space, but as the intersection over union (IOU) which compares the ground truth bounding box for the stop sign with the predicted bounding box. Selecting an appropriate limit on the mAP then involves finding that bound for which an acceptable trade off exists to meet the component level safety requirements. A requirement for the performance of the component will then we specified in terms of the mAP as “*MLR1 -The mean average precision for the component in detecting a stop sign shall be no less than 0.90*”. To justify the sufficiency of our threshold choice it is necessary to show that the three component level safety requirements are satisfied. This requires us to calculate distances from video images, from the development data, and from these show that an IOU that violates the mAP also violates the distance requirement *SR3*.

The ML robustness will then be evaluated by its ability to meet *MLR1* as each of the features of the operational domain are varied within expected bounds. These features are developed through an analysis of the operating domain and an understanding of the conditions which are known, or expected, to impact component performance and the safety requirements as allocated to the component. For our worked example a set of categories are identified to include weather lighting and environmental conditions known to compromise the object detectors ability to correctly identify and classify objects in the image. Within these categories a set of features are then identified, e.g. rain, fog etc. and bounds on expected operational limits created through discussions with domain experts. A set of potential features and dimensions of variability are provided in Table 1. A justification for the sufficiency of this feature set, in the anticipated operational environment, will all be required for the safety argument. having identified our dimensions of variability we can now define a requirement for machine learning robustness as “*MLR2 - The mean average precision defined in MLR1 shall be satisfied over the features and ranges of variability defined in Table 1*”.

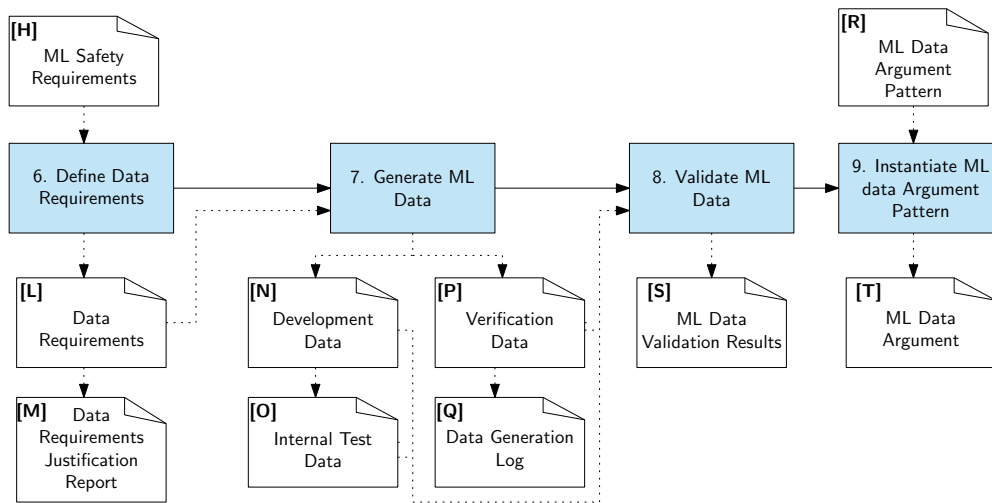


Figure 9: The AMLAS Data Management Assurance Process consisting of 4 process activities and their associated artefacts.

4.3. Data Management Assurance

The data used to train machine learnt components directly impacts their performance. Therefore, the development and assurance of data requirements is a core requirement of any ML assurance argument. Figure 9 shows the third stage of AMLAS which considers these data requirements and consists of 4 activities. During the first activity the data requirements are defined, which may be considered as an encoding of the ML requirements and a data requirements justification report is created.

There is a considerable amount of literature that has been published regarding data management for ML. In previous work [19] we have undertaken an extensive survey of existing literature, and based on this were able to identify four key properties that data should exhibit from an assurance perspective. The specification of the data requirements requires consideration of these four criteria, data relevance (G3.4), data accuracy (G3.6), data balance (G.7), and data completeness (G3.5).

Data relevance refers to the extent to which the development data is representative of the operating environment and architecture of the system into which the ML component will be deployed. Data set reuse between contexts is common, however where existing data sets are used an argument concerning their suitability will need to be provided.

Data accuracy considers the extent to which variations in data gathering, pre-processing and labelling can impact the satisfaction of the ML safety requirements. A requirement for accuracy should therefore define acceptable limits on such variation. This is particularly difficult where labelling requires a level of judgment. In such cases staff training may be an important factor in assuring accuracy.

Data balance typically considers the number of samples for each class present in the data sets. Ideally all data sets used would be perfectly balanced, i.e. the same number of samples would exist for every class of interest. In practice however those samples which are of particular interest in a safety context are often, by their nature, more difficult to obtain. Therefore, requirements for data balance should ensure that a sufficient number of samples exist for each class to ensure the ML safety requirements are satisfied.

While balance considers the number of samples for each class, data completeness concerns how the collected data sets reflect the robustness requirements specified in the ML safety requirements. This will consider the extent to which all of the identified features of concern in the operating domain are present in the data samples for all classes. Ensuring completeness is challenging in open world contexts in which the cross product of the relevant features becomes large.

The data requirements justification report generated in this process will explain why it is believed that the specified requirements are sufficient to develop a model that satisfies the ML component safety requirements.

In order to generate such a report expert review and statistical analysis may be necessary.

Having defined the data requirements in activity 6 we now generate the ML data which meets these requirements. Specifically 3 distinct data sets are generated. Development data is used by the development team to create the

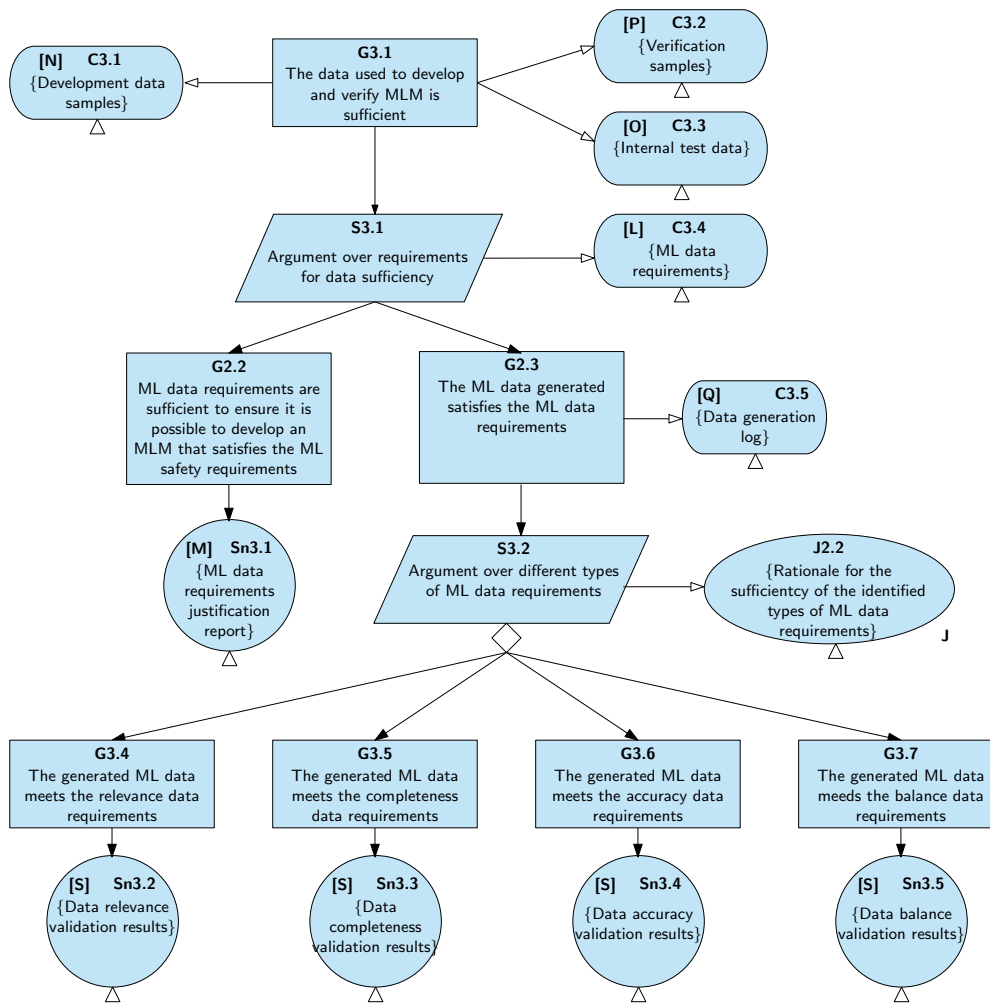


Figure 10: Argument Pattern for ML Data

model and is typically defined to include training and validation data sets. Internal test data is used to assess candidate release models and, where appropriate, may indicate the need for additional development cycles. Only once a model is deemed fit for release by the development team is the model then exposed to the verification data. The generation of the data will typically consider three sub-processes: collection, preprocessing and augmentation [19] the details of which are recorded in the data generation log along with any rationale for the approaches used (simulation, augmentation etc.), analysis undertaken, issues arising, and decisions which necessitate the gathering of specific data samples.

It is crucial that the collection and management of verification data is independent of the model development. The sufficiency of the verification data to assure safety is considered further as part of the model verification stage in section 4.5.

Once the data has been generated the ML data is validated in Activity 8 of the process. The aim of this activity is to ensure that the data are sufficient to meet the data ML requirements. The validation shall consider the gap between the samples obtained and the real-world environment in which the system is to be deployed. The results of the validation as well as any rationale for their sufficiency is then recorded as ML data validation results.

The final activity of this stage then makes use of the argument pattern for the ML data assurance, as shown in Figure 10 and the artefacts generated in the earlier activities to construct an assurance argument for the data management. The top level claim of this argument is that the data used in the development and verification of the model is sufficient and is made for all three of the generated data sets. The argument sets out how this sufficiency can be demonstrated

Table 2: Data distribution by operating domain feature for the use case

Sample ID	Rain				Factor Fog				Glare Angle				...
	1.1	1.2	1.3	1.4	2.1	2.2	2.3	2.4	3.1	3.2	3.3	3.4	
1	✓	-	-	-	✓	-	-	-	✓	-	-	-	...
2	-	✓	-	-	-	✓	-	-	✓	-	-	-	...
3	-	✓	-	-	✓	-	-	-	-	-	-	✓	...
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	...

and makes use of a strategy of goal decomposition into 4 desirable criteria: relevance, completeness, accuracy and balance.

Data Management Assurance for the Use Case

For our motivating example we now consider the four criteria outlined in section 4.3 as they apply to our use case. For data relevance, consideration of the target domain of the vehicle in operation allows us to define requirements for data *relevance* associated with both the system architecture and the operating environment, such as: *DR1: All images used shall be sufficiently representative of images obtained by the camera used by the vehicle in operation*, and *DR2: All road signs in images shall be road signs found on UK roads*.

The requirements on data *accuracy* for this example focus on the creation of labels and bounding boxes in the image data. In this case labelling is carried out using a predominantly manual process and is therefore susceptible to human error, which must be managed, such as: *DR3: All bounding boxes shall be specified such that the entirety of the stop sign is contained within the box irrespective of any obscuration*.

For data balance we define a requirement *DR4: There shall be an equal number of samples for each road sign present in the highway code*.

Finally, for data *completeness* we can define a requirement by considering the dimensions of variability defined as part of the earlier robustness requirement (*MLR2*) such as: *DR5: Data samples shall be obtained that represent each viable combination of features identified in Table 1*.

Having defined the data requirements, data sets can then be created to meet those requirements. This data can be analysed in order to determine if the defined data requirements are met. For example we may say the following in support of the satisfaction of the defined data requirements.

DR1: although a different camera will be used on the target vehicle, the performance characteristics of the camera are comparable and the images obtained are of identical resolution.

DR2: The images were obtained in Germany, so contain some road signs that are not applicable for the vehicle operating on UK roads. This will require the data sets to be checked for images containing irrelevant road signs, which must then be removed.

DR3: All staff undertaking labeling activities were appropriately trained before commencing labelling work. In particular they were trained on the need to individually label partially obscured vehicles. Random sampling from the labelled sets was undertaken to validate the accuracy of the bounding boxes generated.

DR4: Meta data for the data sets indicated that a set of classes were under represented in the data set, i.e. caution signs related to specific types of wildlife. These classes were clustered into a single class to create a balanced data set. In addition there were many more speed restriction signs than necessary and we therefore undersampled from this class to maintain balance.

DR5: The features of the operating environment as defined in Table 1 are mapped to the data sets as shown in Table 2. Analysis of the distribution of features showed that the number of samples with high levels of rain were insufficient. The data set was therefore augmented with synthetic samples to address this shortfall.

4.4. Model Learning Assurance

The model learning assurance process is shown in Figure 11 and consists of 3 activities. The first activity in this stage is the creation of a candidate model which meets the ML safety requirements. Model creation is a highly

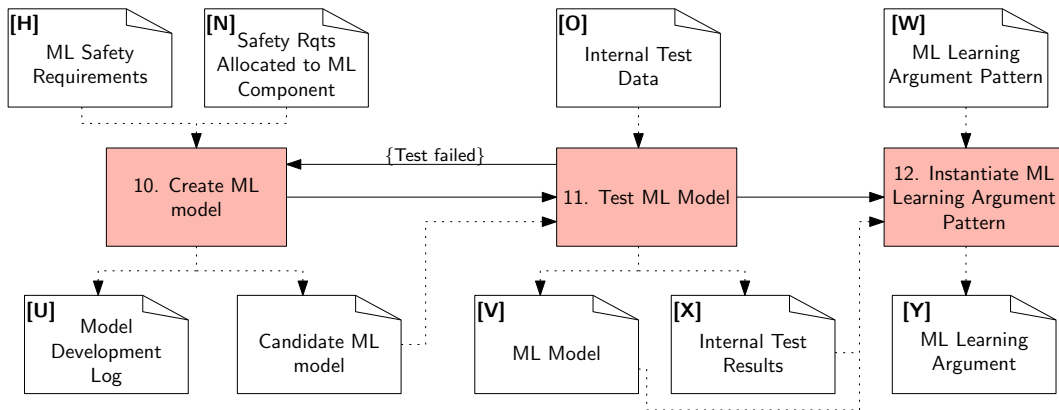


Figure 11: The AMLAS Model Learning Assurance Process consisting of 3 process activities and their associated artefacts.

iterative² process with numerous decision points concerning model structure, learning strategy and hyper-parameter selection. Each of these decision points and the supporting rationale for decisions should be recorded in the model development log. At each development cycle a new candidate model will be created, many of which will lead to new decision points and further iterations of the development cycle. Whilst the focus of AMLAS is on the safety assurance of the machine learnt component we must recognise the practicalities of building components that are performant, robust, reusable and interpretable. As such design decisions will not be purely driven by safety but take account of these wider factors. It should be shown therefore that these decisions do not compromise our ability to satisfy the safety requirements.

Once a candidate model has been created which the development team believe is suitable for deployment it passes to the next activity in the process where testing is undertaken using the internal test data. From the pool of candidate models which meet the safety requirements the ‘optimal’ is selected in light of the, potentially conflicting, requirements taking into account any trade offs which are required for the operating context.

Finally the evidence from the process activities is used with the argument pattern shown in Figure 12 to instantiate an ML learning argument. The central claim of this argument is the sufficiency of the development activities undertaken to create a machine learnt model. This is, in turn, separated into two claims regarding the development approach and the use of the internal test data. For the development approach we argue that the type of model selected, the parameters obtained and the decision taken which drive the development process are appropriate to meet the safety requirements. Each of these claims in turn being supported by the model development log.

Model Learning Assurance for the Use Case

When considering the choice of model to be used in our learning, we select a model from which has been successfully demonstrated in a similar context. For the use case under consideration in this paper we choose YOLO [52] which has been used for perception in autonomous driving applications [53]. This choice is then recorded in the development log with known issues and benefits highlighted as appropriate.

We recognise that training a model is computationally expensive and using existing models as a starting point, can significantly reduce the costs and effort required to obtain a suitable model. We may, therefore, choose to start the development cycle using a deep neural network structure, and transfer learning, from a system previously used in a perception pipeline for an autonomous truck. Our rationale here is that whilst the vehicle for deployment is different to that proposed for our model we believe that the features present in the driving scenario are similar to those present in the new context. Therefore the previously learnt model weights will be sufficiently close to those to be learnt in the new context to allow for faster learning without compromising the safety requirements.

Object detectors are known to be difficult to generalise in open world contexts such as the self-driving vehicle case. Having selected a widely used object detection model (YOLO) we can also apply techniques such as image

²Whilst we omit feedback loops between stages from the diagrams to aid readability, we recognise that the assurance process is likely to be highly iterative in nature with the results of one activity requiring us to return to earlier activities in the process. For example model testing may well highlight the need for additional data generation activities.

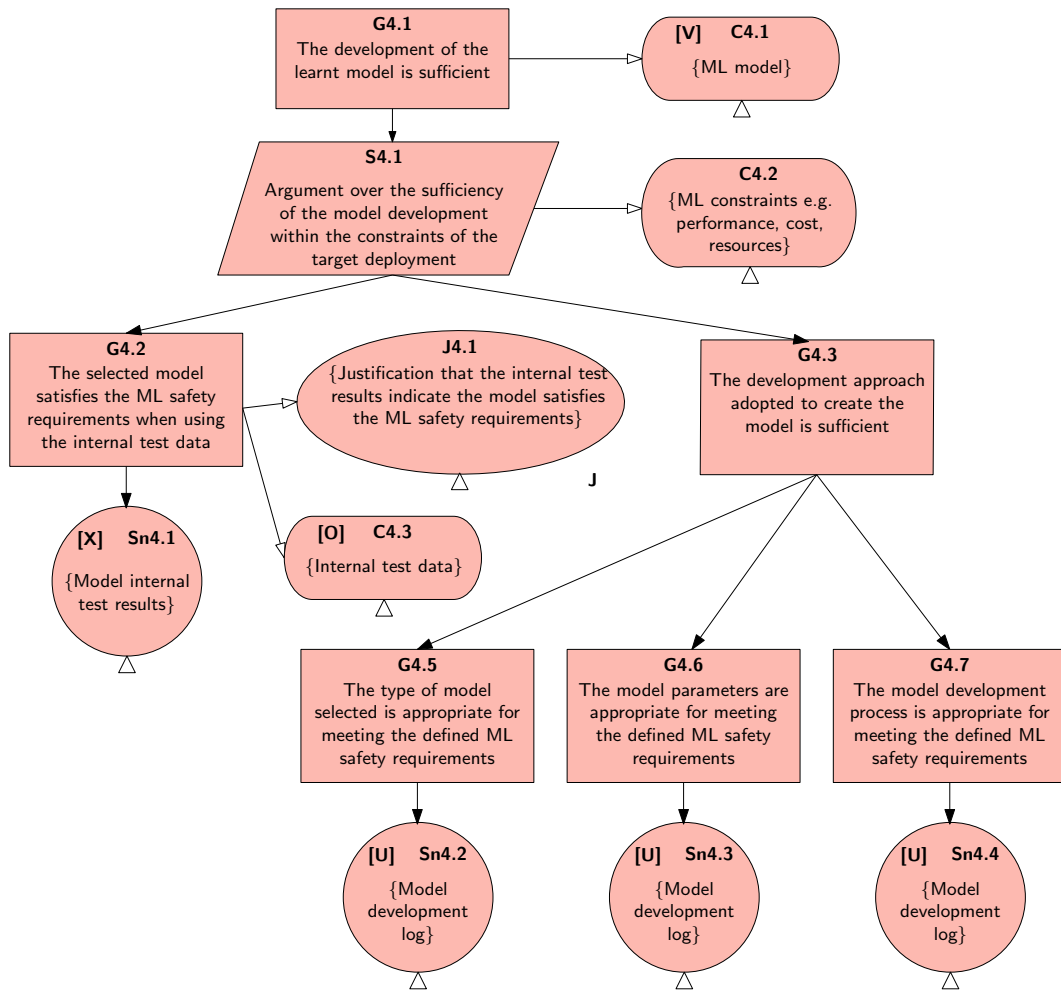


Figure 12: Argument Pattern for Model Learning

mix-up [54] and learning rate schedulers [55] to improve the performance of the model. Once again the choice of training approaches is added to the development log with a justification of selection as well as a record of the improvements in performance observed during training.

It is likely that multiple techniques will be applied to improve the performance of the model based on previous experience of developing these models in a range of contexts. Where a new combination of techniques is employed it is essential to demonstrate that combining these techniques does not compromise the expected efficacy.

In our use case a large number of candidate models are learnt and, given the stochastic nature of model learning, each model is distinct. By applying the internal test data to these models we can identify multiple models which satisfy the defined safety requirements. Model selection therefore requires us to choose from amongst those candidate models. Whilst all models satisfy SR1 (False Negatives) and SR2 (False Positives), we could plot model performance with respect to SR1 and SR2 and select a Pareto optimal model which provides the best trade off between false positives and negative. This requires us to consider the impact on vehicle performance if a stop sign is missed within a single video frame and when a stop sign is identified in error. The justification for model selection is then recorded in the development log.

4.5. Model Verification Assurance

The fifth stage of the AMLAS process, Figure 13, consists of two activities and aims to create an assurance argument for the verification of the ML component. The resulting argument demonstrates that the component will

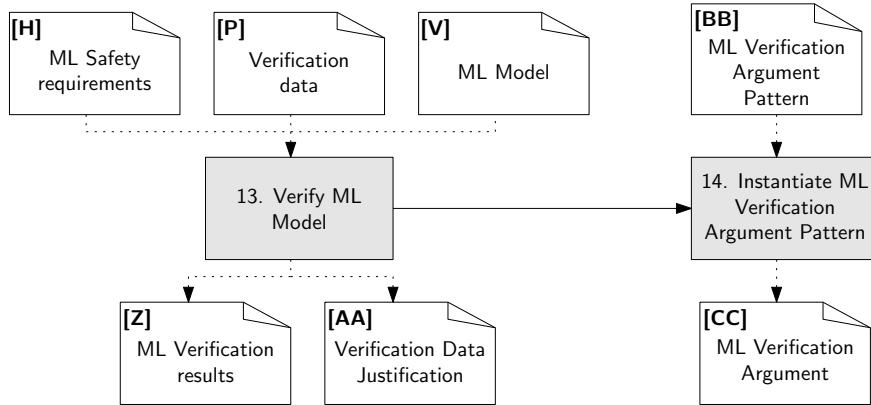


Figure 13: The AMLAS Model Verification Assurance Process consisting of 3 process activities and their associated artefacts.

meet the ML safety requirements when exposed to inputs not present during the development of the model.

This stage of the process starts with model verification which involves both test based and formal verification approaches.

Test based verification combines the model from stage 4 with the verification data from stage 3 to assess whether the model meets the ML safety requirements allocated to the component. This first activity in the stage produces verification results, as well as a verification log, which are then used as evidence in the assurance argument.

One aim of the verification process is to show that the performance of the model with respect to the ML safety requirements, encoded as ML metrics, continue to be met when the model is subjected to data not present in the development process. As such it is imperative that the verification process is independent of development to avoid data leakage [56].

Verification data, like development data, must be accurate and relevant to the specified operating environment. Unlike development data, however, verification data does not need to be balanced or complete. Indeed the data will tend to over represent those cases which are likely to be problematic, but realistic, for the system in context. Verification data should be challenging and gathered using an adversarial mindset within the intent of the system to be developed and be sufficient to verify the intent of the ML safety requirement in the operating context, see Figure 14 (G5.9).

Complete coverage is unlikely to be possible for most real-world systems. It is therefore desirable to find additional forms of verification to support the assurance case. It may be possible, for a subset of cases and ML model structures, to employ formal verification which provides a mathematically rigorous proof that the safety requirements, as specified, are satisfied [57]. This requires us to encode safety requirements as properties amenable to formal verification and this encoding can be difficult. One area where formal verification has proved useful is in providing guarantees on local robustness, that is ensuring that small perturbations in the input space will not lead to changes in the output of the model [58]. These techniques are computationally expensive and, while significant work continues to improve their scalability [59] they are currently not scalable to the size of DNNs that are used in many industrial applications. It is also recognised that such measures of local robustness, whilst mathematically convenient, can lose semantic meaning, which is crucial for safety, and a body of work has therefore arisen to address this issue [60, 48].

The second activity of this stage then instantiates the argument using the pattern shown in Figure 14 and the artefacts generated in the preceding activity. The top level claim in this pattern corresponds to the bottom claim in the safety requirements argument pattern, Figure 8. The claim is supported by the verification activities undertaken as evidence in the verification log. The strategy supporting this claim argues over the verification with respect to the verification data and any formal verification undertaken. For test-based verification we argue that, not only are the test results sufficient, but also that the data and the test platform are sufficient. Similarly for formal verification we do not only argue over the results but also how well they represent the machine learnt model and operating environment.

Model Verification Assurance for the Use Case

For our running example we construct a verification data set which has additional images for environment features

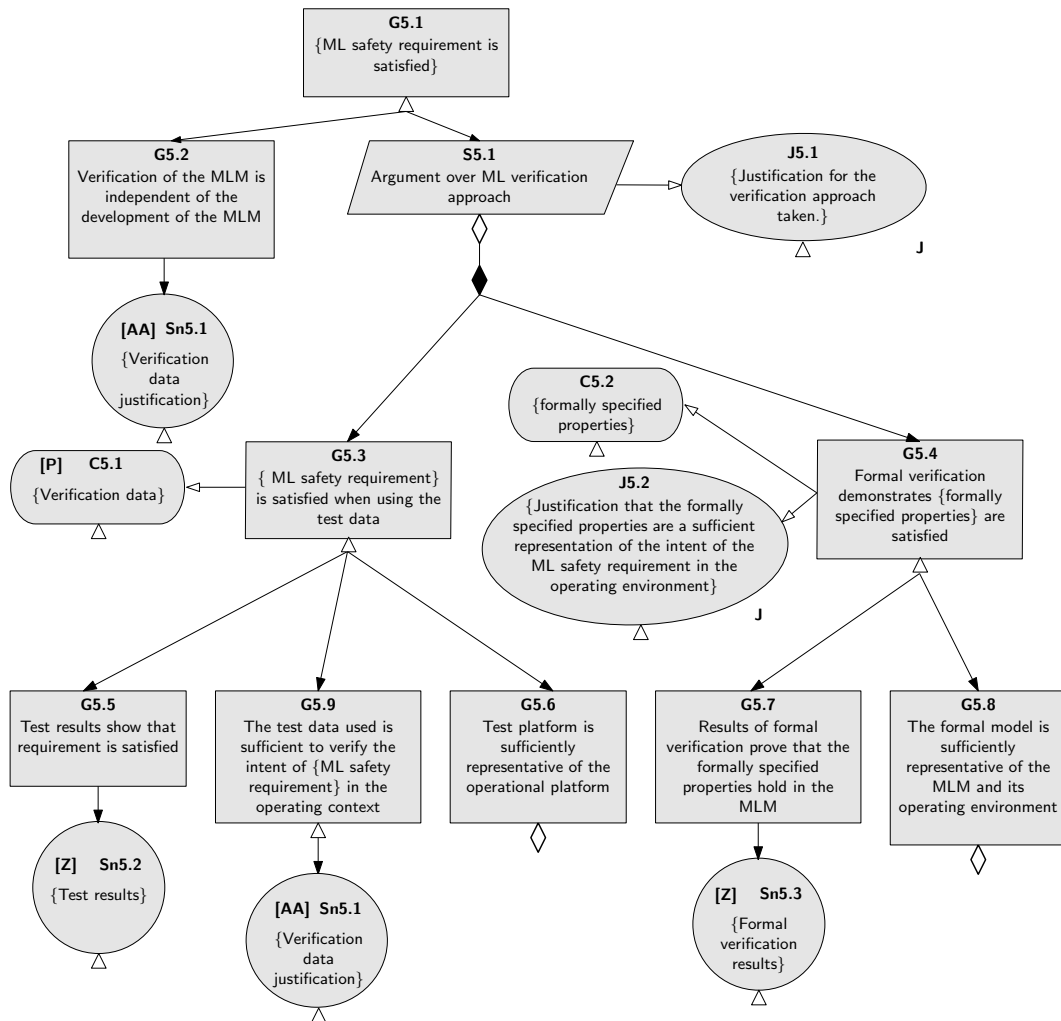


Figure 14: Argument Pattern for Model Verification

at the extremes of the defined variability ranges. Through discussions with domain experts we also add images that contain combinations of features that are known to be particularly challenging for human drivers, or for previously deployed autonomous systems, e.g. fog combined with obscured images in low light. Where it is not possible to obtain real-world images representing such cases image augmentation will be employed to alter existing data samples to provide the required features synthetically.

For our use case fog is particularly challenging given the sensing technology on the platform to which the ML will be deployed. We wish to demonstrate that the ML object detector is no less capable of spotting stop signs in the presence of fog than a highly competent human driver.

To verify this objective we therefore undertake additional tests to identify the level of fog at which the model fails to satisfy the ML safety requirement (MLR1, MLR2)s. This can be achieved for our use case using DeepCert [48], which provides test and formal verification evidence. Formal verification using DeepCert requires us to construct a model of the environmental perturbation which is characterised by a perturbation level ϵ . When this model is composed with the neural network model we are able to verify that for any perturbation of less than ϵ a misclassification will not occur for any sample considered in the verification set. Figure 15 provides an illustrative example of such a plot showing the degradation in mAP as the amount of fog present increases such that MLR1 is violated for a perturbation level ϵ greater than 0.5. If a perturbation level of 0.5 is greater than the levels expected under normal operation

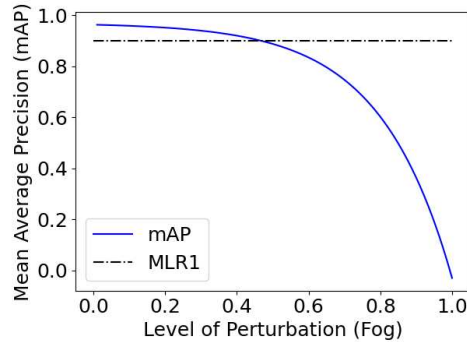


Figure 15: Illustrative example of a robustness plot for environmental perturbations due to fog. The machine learning requirement (MLR1) is violated for perturbations greater than 0.5

then we may argue that the robustness requirements are met.

Since the ML component in our worked example is an object detector, we would also like to verify the robustness of the model with respect to the accuracy of bounding boxes generated, as this is a key determinant of MLR1. In order to undertake formal verification in this case we would use the approach outlined in [61], which allows us to formally verify the model’s robustness to misdetection in the presence of small variation in the input space.

4.6. Model Deployment Assurance

The assurance of model deployment is the final stage of AMLAS and considers the safe integration of the machine learnt component into the target system. Up to this point, the AMLAS process has focused on the assurance of the ML model itself. The aim of this stage is to demonstrate that when the model is deployed as a component as part of the target system the satisfaction of the safety requirements is not compromised. There are a number of safety assurance considerations when integrating an ML component as part of a larger system that are common to the deployment of any safety-related component. This includes activities such as system level testing and safety assessment which are outside of the scope of the AMLAS process. Further guidance on this, as well how the overall system safety case for an autonomous system can be created is provided in [42] [62]. For this stage of AMLAS the focus is on safety assurance issues specifically related to the deployment of an ML component.

Once the ML model has been developed and verified it is then deployed onto the intended hardware and software platform and the broader system of which it is part. Such a deployment may itself require multiple stages with the component first deployed to computational hardware which is then deployed to a subsystem before being integrated with the final hardware platform. This integration process typically involves attaching the components to its inputs, such as the systems sensing devices, and pre-processing pipelines as well as connecting the outputs to traditional software units which ultimately produce system outputs.

The assurance of the ML deployment, as shown in Figure 16, consists of four activities. This assurance process is not only applicable to initial deployment but to any subsequent deployment when component updates are applied.

An important part of this integration is to identify what the deployment assumptions are that could impact the safety of the ML component if they are violated. In the first activity, these assumptions are derived from the system description that was specified at the start of the AMLAS process and are recorded in a deployment assumptions log. The deployment assumptions should consider the hardware upon which the model will execute (such as AI edge devices), the nature of the system into which the ML component will be integrated (such as the type of sensors that are used); and supporting software libraries which may be different to that used at development time (such as TensorFlow Lite [63] or PyTorch Mobile [64]).

It is important to also consider the assurance of the ML component through-life in evolving open environments. This requires us to record assumptions concerning the input distribution likely to be encountered by the model in operation and the strategies for identifying when the ML component is being used outside of its defined safe scope of operation. Identifying these environmental deviations requires the creation of monitors which are aligned with the

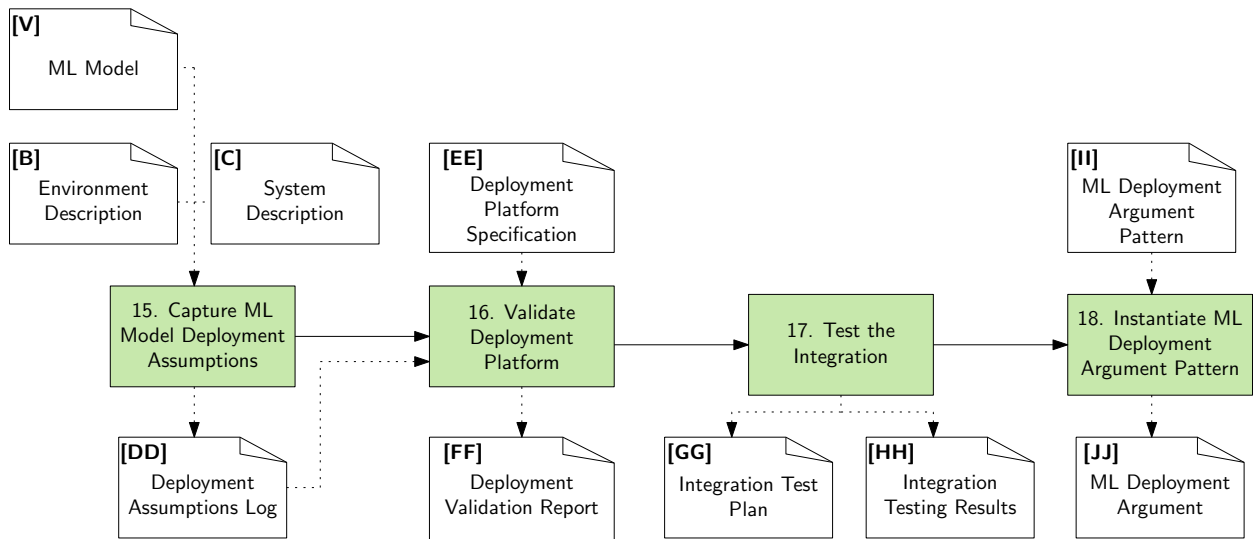


Figure 16: The AMLAS Model Deployment Assurance Process consisting of 4 process activities and their associated artefacts.

description of the operational environment, further emphasising the need for a complete description of the operational environment. Further work developing a framework for the through-life assurance of ML components is ongoing [65].

We need to firstly validate that the deployment hardware and software is as assumed in the deployment assumptions log. Where variations exist these must be shown not to compromise the satisfaction of the system safety requirements, for example where a newer version of a software library is to be used, sufficient verification has been provided to assure backwards compatibility.

Once the ML component is integrated with the deployment platform, an integration test plan should be created to check that the performance of the model is not adversely affected by the deployment platform. The test plan should define the tests to be conducted, the expected outcomes, and a justification of the sufficiency of the tests to demonstrate that the performance of the deployed model continues to satisfy the safety requirements. This test plan is then executed and the integration test results recorded.

The final activity is then to instantiate the ML deployment argument using the pattern as shown in Figure 17. The primary goal of showing that the deployment of the ML component satisfies the safety requirements is addressed by arguing over the validation of the deployment platform, integration testing and through-life monitoring.

With relation to the monitoring of the ML component, the Deployment Assumption Log can be used as evidence to demonstrate that the monitoring requirements are complete and correct. The integration testing argument demonstrates that the integration test are passed and that the testing performed is sufficient to test both the assumptions and the effects of the hardware deployment.

It is important to emphasise the key role of through-life assurance, and the role played by monitoring, as captured in the argument strategy S6.1: *‘Argument over testing and monitoring of the deployed ML component’*. Monitoring here should be interpreted in two complementary senses: online and offline.

Online monitoring continuously evaluates the performance of the ML component in actual use, at run-time, against its predefined safety requirements (Stage 2 of AMLAS). If these requirements are not met, the wider system’s design must incorporate robustness recovery mechanisms, such as switching to an alternative component or operational mode, depending on the risk posed by the identified violations. For example, Denney and Pai propose a dynamic safety assurance framework for ML-based and autonomous systems [66]. They demonstrate its implementation for autonomous taxiing of uncrewed systems utilising deep convolutional neural networks and derive monitoring indicators and metrics, and define corresponding recovery mechanisms. This work ultimately leads to an operationalisation of the notion of dynamic safety cases, with clear links to online monitoring in Stage 6 of AMLAS.

Offline monitoring, conversely, addresses broader changes within the operating and organisational environment that could challenge the ongoing suitability of the safety requirements themselves. For instance, a recent application

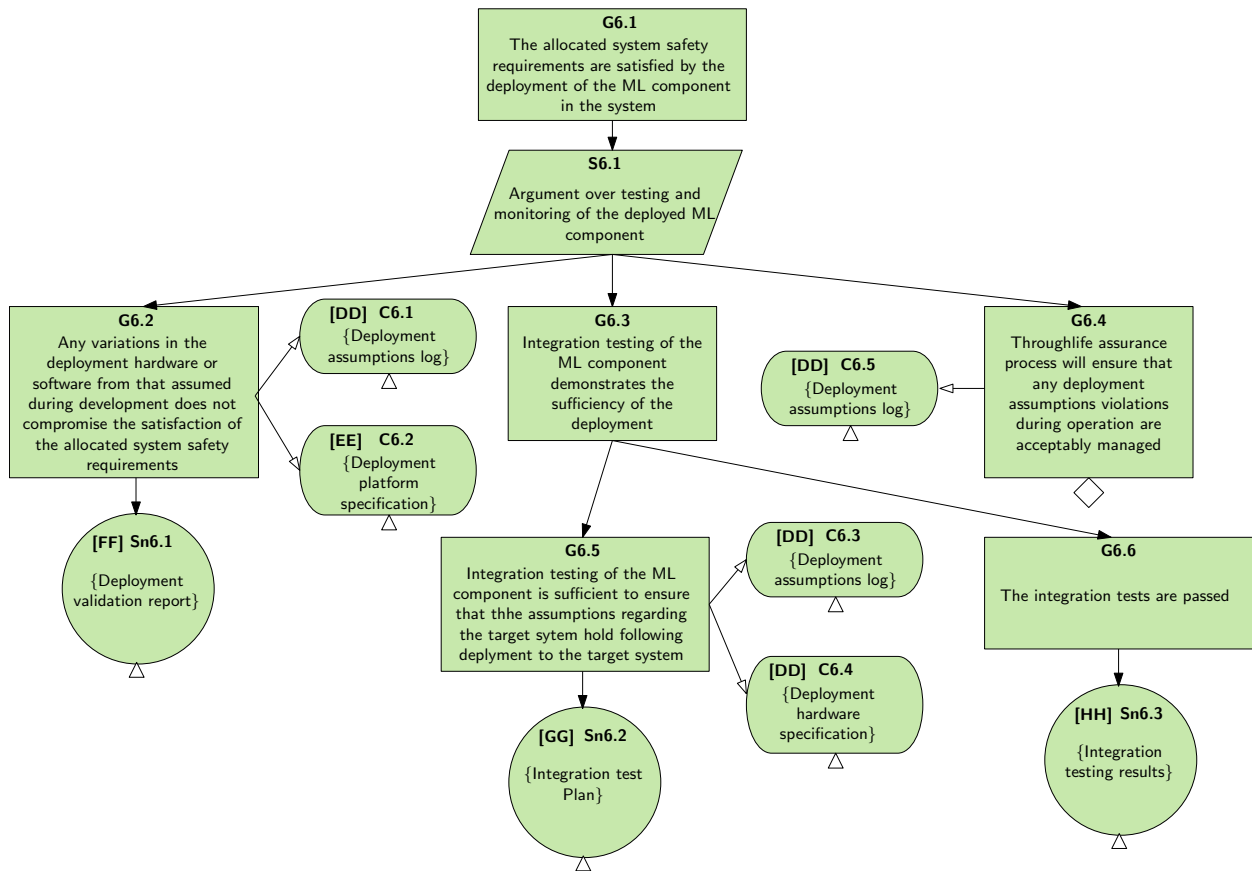


Figure 17: Argument Pattern for Model Deployment

of AMLAS to an AI-decision support system for prostate cancer diagnosis, currently undergoing clinical trials and in clinical use, highlighted emerging issues during deployment [66]. These triggered a re-evaluation of the initial hazard analysis results, linked to AMLAS Stage 1, due to the emergence of new hazardous conditions not identified during the design stages (e.g., “True positive classification but under-annotated tissue map”).

What this reinforces is that the deployment stage is intrinsically linked with essential system-level and non-ML specific obligations [67] [68]. These include architectural dependability mechanisms for redundancy and recovery, as well as established procedures and organisational activities that ensure hazard analysis and risk assessment are conducted throughout the system’s lifecycle, rather than being limited to initial development and pre-deployment phases. This whole-system, through-life, approach represents best practice in safety engineering and must be maintained, and where appropriate adapted (given the higher frequency of updates), for ML-based systems.

Model Deployment Assurance for the Use Case

For our running example we consider the situation where the hardware to which the model will be deployed is validated to be identical to the edge computing device initially specified in the deployment assumptions log. This hardware, however, requires a “lightweight” ML library which differs from that used at development time. In the case of the running example we may assume that TensorFlow [63] is used for development whilst TensorFlow Lite [63] is required for deployment. TensorFlow Lite requires us to optimise the model using an inference compiler [69] which will quantize model weights [70]. The deployment model may have reduced performance due to the need to store weights at a lower resolution. We therefore need to demonstrate that the performance of the degraded model is sufficient such that the allocated system safety requirements (SR1, SR2, SR3) can still be satisfied. This is done through integration testing in which we rerun verification activities on the deployment hardware.

5. Application of Methodology

The practical aim of AMLAS is to provide a systematic methodology for assuring the safety of ML components when used in safety-critical systems, especially those with autonomous capabilities. So far, AMLAS has been applied in several domains, including healthcare [71][66], automotive [72] and aerospace [73]. It has also supported policy and industry guidance in defence³, healthcare⁴ and automotive⁵.

For example, Festor et al applied AMLAS for the assurance of the ‘AI Clinician’ [71], a clinical decision support system for the treatment of sepsis. The treatment of sepsis in ICU includes the administration of intravenous fluids and/or vasopressors. The AI Clinician system uses Reinforcement Learning (RL) for recommending doses for either intravenous fluids or vasopressors. Determining the scope of safety assurance and defining the ML safety requirements presented a significant challenge (i.e. stages 1 and 2 of AMLAS). This was due to the complex and highly uncertain nature of the healthcare setting and the clinical condition. This required close collaboration between the clinicians, ML developers and safety engineers who worked together on defining unsafe treatment scenarios and importantly on determining how the outputs of the RL model could potentially contribute to these scenarios. Using AMLAS, the control of this clinical hazard was refined into specific safety requirements and constraints, including at the model learning stage.

Borg et al [27] reported similar results of applying AMLAS for the safety assurance of an ML-based pedestrian automatic emergency braking system. The authors also highlighted the importance of context and safety requirements, especially when combined with further evaluation through simulation. They emphasised the significance of the iterative nature of ML assurance and how this could be applied systematically by following the AMLAS process.

Hawkins et al [73] applied AMLAS in full to an ML-based wildfire alert system. The authors provided a complete snapshot of an ML safety argument that centred on the management of two overarching hazards (i.e. missing a real emergency or reporting a false one) and explained how they were controlled at the different stages of AMLAS.

It is fundamental principle that safety assurance is a system-level issue. This is a strength of AMLAS by insisting on traceability between system hazards, overall safety requirements and ML safety requirements. The weakness lies in the lack of sufficient consideration of safety as a whole system property in ML development and practice generally, due to either lack of conceptual system-level clarity (e.g. boundaries between systems, services and pathways in healthcare are rarely defined [74]) or overemphasis on technology-driven solutions and investment (i.e. asking questions about the safety of the technology rather than the safety of the system of which the technology is one of a number of interacting and interdependent components). As such, to provide more complete assurance, AMLAS has to be combined with other guidelines and standards that consider the safety of the wider system and context. This could be domain independent such as the *Guidance on the Safety Assurance of Autonomous Systems in Complex Environments (SACE)* [42] or sector specific such as the *ISO 21448 standard (Road vehicles—Safety of the intended functionality)* [75].

It is important to acknowledge that this paper has focused on the conceptual and methodological aspects of AMLAS, supported by an illustrative example. As indicated above, existing and publicly available studies show some evidence of applicability of the use of ML in various sectors, most notably against supervised learning technologies. We are currently evaluating AMLAS further using research case studies. These will provide qualitative insights into the methodology’s feasibility and its effectiveness in generating a clear safety case for ML, as well as evaluating how it fits within a wider system safety case. We are encouraged by the independent studies reported in the literature, such as the AMLAS safety case developed by Borg et al [27], that critically evaluate AMLAS and generate impartial evidence of its potential benefits and limitations. A number of key, open, questions remain, including: how precise metrics for ML safety requirements can be defined; how the data desiderata could be justified from a safety perspective; and how to develop monitoring mechanisms to ensure that the ML safety requirements and verification evidence remain valid through life.

³Defence Science and Technology Laboratory: <https://www.gov.uk/government/publications/assurance-of-ai-and-autonomous-systems-a-dstl-biscuit-book/assurance-of-artificial-intelligence-and-autonomous-systems-a-dstl-biscuit-book>

⁴English National Health Service: <https://digital.nhs.uk/services/clinical-safety/documentation/healthcare-supplementary-guidance-for-amlas>

⁵SO/CD PAS 8800 Road Vehicles Safety and artificial intelligence: <https://www.iso.org/standard/83303.html>

6. Related Work

In this section, we provide an overview of significant research efforts and developments in this field, focusing on surveys and safety assurance frameworks rather than specific verification and validation (V&V) methods.

Dong et al [76] presented a safety case template for machine learning components employed as part of a system, with an emphasis on quantitative aspects combining HAZOP and quantitative FTA to derive probabilities of basic events associated with the ML components. Burton et al [77] presented a structure for a safety assurance argument for a safety-relevant function implemented using supervised ML and addressed the impact of uncertainty on the confidence in the safety argument. Bloomfield et al [78] proposed safety assurance argument templates for autonomous systems that include ML components. Specifically, they focused on the requirements for an ML component and presented a safety monitor architecture for an autonomous system including at least one ML component. What each of these papers failed to provide was guidance on how to link the safety assurance activities and ML development processes to the proposed safety case structures.

In contrast other work has focused on specific aspects of an ML safety lifecycle. For example Alves et al [79] and Koopman [80] identified the challenges in safety assurance of increasingly autonomous systems, specifically discussing the issues in V&V of ML techniques and explored the approaches for developing assured ML systems, e.g. anomaly detection of the inputs to the ML-based components. Burton et al [81] introduced a contract-based approach to satisfy a set of safety-related requirements for ML components under a given set of assumptions, with the aim of supporting modular V&V of systems including ML-based components.

There are a number of dedicated survey papers of methods for safety assurance of ML-based systems. A review by Schwalbe and Schels [82], carried out in the context of autonomous driving, provided a summary table categorising methods against requirements, the ML development process and V&V of ML. Other relevant surveys of techniques and methods include Ashmore et al. [19] which provided the initial basis for the development of AMLAS.

There has also been work looking at safety standards for ML-based systems. Salay et al [83] conducted a complete applicability assessment of the automotive standard ISO 26262 for ML-based software and recommended explicitly addressing the ML lifecycle with customised tools and techniques that can be applied to this lifecycle. There is also ongoing work on ML-focused safety standards, e.g., ISO PAS 8800 is considering V&V of ML-based components in automotive systems [84].

Some other research has explicitly explored the technical aspects of the safety of ML models. For example, Jia et al [85] have illustrated how to use safety analysis methods to shape loss functions for deep reinforcement learning. Rudin has argued that we should develop more interpretable deep learning models, e.g. [86],[87], for use in high-stakes applications, including where safety is a concern [88]. Other work has focussed on uncertainty of ML, e.g. [89] and [90].

We can see from this review that there is substantial interest in the use of safety cases for safety assurance of ML systems, with Dong et al [76] concluding that safety cases are now “the emerging consensus within both, industry and academia”. Most of the literature focuses on part of the issues, e.g. deriving safety requirements for ML components, data quality or V&V for ML models etc. Many papers are also focused on specific application sectors, most commonly autonomous driving. Some of the approaches that discuss ML are also presented at a whole system level and do not go into the details of the ML-based elements of the system, e.g. [78], [79], and [91].

In summary, none of the existing research has developed a domain-independent methodology that considers the whole of the ML life-cycle and describes how a resulting safety case can be developed for the ML component. We believe that this is the first work to provide such an approach.

7. Conclusions and Future Work

In this paper, we introduced the six-stage AMLAS process, which enables the safe incorporation of machine-learned components into safety-critical autonomous systems. AMLAS aligns with the machine learning lifecycle, proactively integrating safety assurance into ML development. When combined with the provided safety argument patterns, AMLAS supports the creation of a compelling safety case for the ML component. We described the application of AMLAS in an automotive context and showed how AMLAS is being used across various application domains to demonstrate the safety of ML components.

Future research directions for this work include applying AMLAS to larger scale, real-world deployments which will necessitate the development of guidance and practical toolsets for both developers and safety engineers which account for domain-specific concerns. Currently, AMLAS primarily focuses on ML technologies that use supervised learning in perception and prediction tasks. Further work is needed to extend and adapt AMLAS for other forms of ML, including reinforcement learning, unsupervised tasks, and emerging technologies such as graph neural networks and neurosymbolic reasoning used in autonomous decision-making frameworks. We also recognise the desire to use autonomous systems in highly dynamic environments and the challenges which this presents for ensuring the safety of ML components is maintained throughout system operation. Therefore, we plan to extend AMLAS to provide continuous safety assurance throughout the system’s life.

References

- [1] M. Nagendran, Y. Chen, C. A. Lovejoy, A. C. Gordon, M. Komorowski, H. Harvey, E. J. Topol, J. P. Ioannidis, G. S. Collins, M. Maruthappu, Artificial intelligence versus clinicians: systematic review of design, reporting standards, and claims of deep learning studies, *BMJ* 368 (2020).
- [2] N. Rank, B. Pfahringer, J. Kempfert, C. Stamm, T. Kühne, F. Schoenrath, V. Falk, C. Eickhoff, A. Meyer, Deep-learning-based real-time prediction of acute kidney injury outperforms human predictive performance, *NPJ digital medicine* 3 (2020) 139.
- [3] A. le Calvez, D. Cliff, Deep learning can replicate adaptive traders in a limit-order-book financial market, in: 2018 IEEE Symposium Series on Computational Intelligence (SSCI), IEEE, 2018, pp. 1876–1883.
- [4] B. S. Miller, S. Madhusudhana, M. G. Aulich, N. Kelly, Deep learning algorithm outperforms experienced human observer at detection of blue whale D-calls: a double-observer analysis, *Remote Sensing in Ecology and Conservation* 9 (2023) 104–116.
- [5] Y. Cui, R. Chen, W. Chu, L. Chen, D. Tian, Y. Li, D. Cao, Deep learning for image and point cloud fusion in autonomous driving: A review, *IEEE Transactions on Intelligent Transportation Systems* 23 (2021) 722–739.
- [6] M. Chelouati, A. Boussif, J. Beugin, E.-M. El Koursi, Graphical safety assurance case using Goal Structuring Notation (GSN) — challenges, opportunities and a framework for autonomous trains, *Reliability Engineering & System Safety* 230 (2023) 108933. URL: <https://www.sciencedirect.com/science/article/pii/S0951832022005488>. doi:10.1016/j.res.2022.108933.
- [7] M. Gyllenhammar, G. R. de Campos, M. Törngren, The road to safe automated driving systems: A review of methods providing safety evidence, *IEEE Transactions on Intelligent Transportation Systems* (2025).
- [8] R. Rai, M. K. Tiwari, D. Ivanov, A. Dolgui, Machine learning in manufacturing and industry 4.0 applications, 2021.
- [9] H. A. Gohel, H. Upadhyay, L. Lagos, K. Cooper, A. Sanzetea, Predictive maintenance architecture development for nuclear infrastructure using machine learning, *Nuclear Engineering and Technology* 52 (2020) 1436–1442.
- [10] P. V. Gunckel, G. Lobos, F. K. Rodríguez, R. M. Bustos, D. Godoy, Methodology Proposal for the Development of Failure Prediction Models Applied to conveyor belts of Mining Material using Machine Learning, *Reliability Engineering & System Safety* (2024) 110709. URL: <https://www.sciencedirect.com/science/article/pii/S0951832024007804>. doi:10.1016/j.res.2024.110709.
- [11] M. Hussain, T. Zhang, Machine learning-based outlier detection for pipeline in-line inspection data, *Reliability Engineering & System Safety* 254 (2025) 110553. URL: <https://www.sciencedirect.com/science/article/pii/S0951832024006252>. doi:10.1016/j.res.2024.110553.
- [12] M. Shiokari, H. Itoh, T. Yuzui, E. Ishimura, R. Miyake, J. Kudo, S. Kawashima, Structure model-based hazard identification method for autonomous ships, *Reliability Engineering & System Safety* 247 (2024) 110046. URL: <https://www.sciencedirect.com/science/article/pii/S0951832024001212>. doi:10.1016/j.res.2024.110046.
- [13] T. Johansen, S. Blindheim, T. R. Torben, I. B. Utne, T. A. Johansen, A. J. Sørensen, Development and testing of a risk-based control system for autonomous ships, *Reliability Engineering & System Safety* 234 (2023) 109195. URL: <https://www.sciencedirect.com/science/article/pii/S0951832023001102>. doi:10.1016/j.res.2023.109195.
- [14] E. Chen, H. Bao, N. Dinh, Evaluating the reliability of machine-learning-based predictions used in nuclear power plant instrumentation and control systems, *Reliability Engineering & System Safety* 250 (2024) 110266. URL: <https://www.sciencedirect.com/science/article/pii/S0951832024003387>. doi:10.1016/j.res.2024.110266.
- [15] A. Linja, T. I. Mamun, S. T. Mueller, When Self-Driving Fails: Evaluating Social Media Posts Regarding Problems and Misconceptions about Tesla’s FSD Mode, *Multimodal Technologies and Interaction* 6 (2022) 86.
- [16] G. Varoquaux, V. Cheplygina, Machine learning for medical imaging: methodological failures and recommendations for the future, *NPJ digital medicine* 5 (2022) 48.
- [17] Z. Xu, J. H. Saleh, Machine learning for reliability engineering and safety applications: Review of current status and future opportunities, *Reliability Engineering & System Safety* 211 (2021) 107530. URL: <https://www.sciencedirect.com/science/article/pii/S0951832021000892>. doi:10.1016/j.res.2021.107530.
- [18] A. Serban, K. van der Blom, H. Hoos, J. Visser, Software engineering practices for machine learning — Adoption, effects, and team assessment, *Journal of Systems and Software* 209 (2024) 111907. URL: <https://www.sciencedirect.com/science/article/pii/S0164121223003023>. doi:10.1016/j.jss.2023.111907.
- [19] R. Ashmore, R. Calinescu, C. Paterson, Assuring the machine learning lifecycle: Desiderata, methods, and challenges, *ACM Computing Surveys (CSUR)* 54 (2021) 1–39.
- [20] S. P. Wilson, T. P. Kelly, J. A. McDermid, Safety case development: Current practice, future prospects, in: *Safety and Reliability of Software Based Systems: Twelfth Annual CSR Workshop (Bruges, 12–15 September 1995)*, Springer, 1997, pp. 135–156.
- [21] P. Bishop, R. Bloomfield, A methodology for safety case development, in: *Safety and Reliability*, volume 20.1, Taylor & Francis, 2000, pp. 34–42.

- [22] N. G. Leveson, The use of safety cases in certification and regulation, in: ESD Working Paper Series, Massachusetts Institute of Technology, Engineering Systems Division, 2011.
- [23] N. Blackwell, S. Leinster-Evans, S. K. Dawkins, Developing safety cases for integrated flight systems, in: 1999 IEEE Aerospace Conference. Proceedings (Cat. No. 99TH8403), volume 5, IEEE, 1999, pp. 225–240.
- [24] R. Wang, J. Guiochet, G. Motet, W. Schön, Modelling confidence in railway safety case, *Safety Science* 110 (2018) 286–299.
- [25] Ministry of Defence, Defence Standard 00-56 Issue 4: Safety Management Requirements for Defence Systems (2007).
- [26] P. Oy, Safety case for the disposal of spent nuclear fuel at Olkiluoto, <https://www.osti.gov/etdweb/servlets/purl/22134704>, 2012. Accessed: 2-2-2024.
- [27] J. Birch, R. Rivett, I. Habli, B. Bradshaw, J. Botham, D. Higham, P. Jesty, H. Monkhouse, R. Palin, Safety cases and their role in iso 26262 functional safety assessment, in: Computer Safety, Reliability, and Security: 32nd International Conference, SAFECOMP 2013, Toulouse, France, September 24-27, 2013. Proceedings 32, Springer, 2013, pp. 154–165.
- [28] NHS Digital Clinical Safety team, DCB0160: Clinical Risk Management: its Application in the Deployment and Use of Health IT Systems, <https://digital.nhs.uk/data-and-information/information-standards/information-standards-and-data-collections-including-extractions/publications-and-notifications/standards-and-collections/dcb0160-clinical-risk-management-its-application-in-the-deployment-and-use-of-health-it-systems>, 2020. Accessed: 04-04-2023.
- [29] T. P. Kelly, et al., Arguing safety: a systematic approach to managing safety cases, Ph.D. thesis, University of York York, UK, 1999.
- [30] UK Office for Nuclear Regulation, The purpose, scope, and content of safety cases, rev. 4, http://www.onr.org.uk/operational/tech_asst_guides/ns-tast-gd-051.pdf, 2019.
- [31] US Dept. Health and Human Services, Infusion pumps total product life cycle—guidance for industry & FDA staff, <http://www.fda.gov/downloads/medicaldevices/deviceregulationandguidance/guidancedocuments/ucm209337.pdf>, 2014. Accessed: 23-10-2022.
- [32] European Organisation for the Safety of Air Navigation, Safety case development manual edition 2.2, <https://www.eurocontrol.int/sites/default/files/article/content/documents/nm/link2000/safety-casedevelopment-manual-v2.2-ri-13nov06.pdf>, 2006. Accessed: 23-10-2022.
- [33] The Assurance Case Working Group (ACWG), Goal structuring notation standard, version 2, <https://scsc.uk/r141B:1?t=1>, 2018. Accessed: 23-10-2022.
- [34] J. Spriggs, GSN-the goal structuring notation: A structured approach to presenting arguments, Springer Science & Business Media, 2012.
- [35] T. P. Kelly, J. A. McDermid, Safety case construction and reuse using patterns, in: *Safe Comp* 97, Springer, 1997, pp. 55–69.
- [36] Y. Li, H. Wang, L. M. Dang, T. N. Nguyen, D. Han, A. Lee, I. Jang, H. Moon, A deep learning-based hybrid framework for object detection and recognition in autonomous driving, *IEEE Access* 8 (2020) 194228–194239.
- [37] D. Barros, J. C. Moura, C. R. Freire, A. C. Taleb, R. A. Valentim, P. S. Morais, Machine learning applied to retinal image processing for glaucoma detection: review and perspective, *Biomedical engineering online* 19 (2020) 1–21.
- [38] I. Goodfellow, Y. Bengio, A. Courville, Deep learning, MIT press, 2016.
- [39] S. Liu, L. Li, J. Tang, S. Wu, J.-L. Gaudiot, Creating autonomous vehicle systems, *Synthesis Lectures on Computer Science* 8 (2020) i–216.
- [40] H. E. Monkhouse, I. Habli, J. McDermid, An enhanced vehicle control model for assessing highly automated driving safety, *Reliability Engineering & System Safety* 202 (2020) 107061.
- [41] N. G. Leveson, *Safeware: system safety and computers*, ACM, 1995.
- [42] R. Hawkins, M. Osborne, M. Parsons, M. Nicholson, J. McDermid, I. Habli, Guidance on the Safety Assurance of Autonomous Systems in Complex Environments (SACE), arXiv preprint arXiv:2208.00853 (2022).
- [43] International Standards Organization, PAS 1883:2020, Operational Design Domain (ODD) taxonomy for an automated driving system (ADS) – Specification, 2020.
- [44] A. Shakeri, Formalization of operational domain and operational design domain for automated vehicles, in: 2024 IEEE 24th International Conference on Software Quality, Reliability, and Security Companion (QRS-C), IEEE, 2024, pp. 990–997.
- [45] P. Weissensteiner, G. Stettinger, S. Khastgir, D. Watzeng, Operational design domain-driven coverage for the safety argumentation of automated vehicles, *IEEE Access* 11 (2023) 12263–12284.
- [46] M. Rahimi, J. Guo, S. Kokaly, M. Chechik, Toward requirements specification for machine-learned components, in: 2019 IEEE 27th International Requirements Engineering Conference Workshops (REW, IEEE, 2019, p. 241–244.
- [47] J. De Fauw, J. R. Ledsam, B. Romera-Paredes, S. Nikolov, N. Tomasev, S. Blackwell, H. Askham, X. Glorot, B. O’Donoghue, D. Visentin, et al., Clinically applicable deep learning for diagnosis and referral in retinal disease, *Nature medicine* 24 (2018) 1342–1350.
- [48] C. Paterson, H. Wu, J. Grese, R. Calinescu, C. S. Păsăreanu, C. Barrett, DeepCert: Verification of contextually relevant robustness for neural network image classifiers, in: International Conference on Computer Safety, Reliability, and Security, Springer, 2021, pp. 3–17.
- [49] R. Calinescu, N. Alasmari, M. Gleirscher, Maintaining driver attentiveness in shared-control autonomous driving, in: 2021 International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS), IEEE, 2021, pp. 90–96.
- [50] J. Redmon, S. Divvala, R. Girshick, A. Farhadi, You only look once: Unified, real-time object detection, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 779–788.
- [51] P. Henderson, V. Ferrari, End-to-end training of object class detectors for mean average precision, in: Computer Vision–ACCV 2016: 13th Asian Conference on Computer Vision, Taipei, Taiwan, November 20-24, 2016, Revised Selected Papers, Part V 13, Springer, 2017, pp. 198–213.
- [52] J. Redmon, S. Divvala, R. Girshick, A. Farhadi, You Only Look Once: Unified, Real-Time Object Detection, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 779–788. doi:10.1109/CVPR.2016.91.
- [53] A. Sarda, S. Dixit, A. Bhan, Object detection for autonomous driving using YOLO [you only look once] algorithm, in: 2021 Third international conference on intelligent communication technologies and virtual mobile networks (ICICV), IEEE, 2021, pp. 1370–1374.
- [54] H. Zhang, M. Cisse, Y. N. Dauphin, D. Lopez-Paz, mixup: Beyond empirical risk minimization, arXiv preprint arXiv:1710.09412 (2017).
- [55] Z. Zhang, T. He, H. Zhang, Z. Zhang, J. Xie, M. Li, Bag of freebies for training object detection neural networks, arXiv preprint

- arXiv:1902.04103 (2019).
- [56] S. Kaufman, S. Rosset, C. Perlich, O. Stitelman, Leakage in data mining: Formulation, detection, and avoidance, *ACM Transactions on Knowledge Discovery from Data (TKDD)* 6 (2012) 1–21.
 - [57] H. Wu, O. Isac, A. Zeljić, T. Tagomori, M. Daggitt, W. Kokke, I. Refaeli, G. Amir, K. Julian, S. Bassan, et al., Marabou 2.0: A versatile formal analyzer of neural networks, arXiv preprint arXiv:2401.14461 (2024).
 - [58] G. Katz, C. Barrett, D. L. Dill, K. Julian, M. J. Kochenderfer, Reluplex: An efficient SMT solver for verifying deep neural networks, in: *Computer Aided Verification: 29th International Conference, CAV 2017, Heidelberg, Germany, July 24–28, 2017, Proceedings, Part I* 30, Springer, 2017, pp. 97–117.
 - [59] C. Brix, S. Bak, T. T. Johnson, H. Wu, The Fifth International Verification of Neural Networks Competition (VNN-COMP 2024): Summary and Results, arXiv preprint arXiv:2412.19985 (2024).
 - [60] C. R. Serrano, P. M. Sylla, M. A. Warren, Generate and verify: Semantically meaningful formal analysis of neural network perception systems, arXiv preprint arXiv:2012.09313 (2020).
 - [61] A. Raviv, Y. Y. Elboher, M. Aluf-Medina, Y. L. Weiss, O. Cohen, R. Assa, G. Katz, H. Kugler, Formal verification of object detection, arXiv preprint arXiv:2407.01295 (2024).
 - [62] I. Habli, R. Hawkins, C. Paterson, P. Ryan, Y. Jia, M. Sujan, J. McDermid, The big argument for ai safety cases, arXiv preprint arXiv:2503.11705 (2025).
 - [63] M. A. et al, TensorFlow: Large-scale machine learning on heterogeneous systems, <https://www.tensorflow.org/>, 2015. Accessed: 2-2-2024.
 - [64] A. e. a. Paszke, PyTorch: An imperative style, high-performance deep learning library, in: *Advances in Neural Information Processing Systems 32*, Curran Associates, Inc., 2019, pp. 8024–8035. URL: <http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf>.
 - [65] C. Picardi, R. D. Hawkins, C. Paterson, I. Habli, Transfer assurance for machine learning in autonomous systems, in: *Proceedings of the Workshop on Artificial Intelligence Safety (SafeAI 2023)*, CEUR Workshop Proceedings, 2023.
 - [66] Y. Jia, C. Verrill, K. White, M. Dolton, M. Horton, M. Jafferji, I. Habli, A deployment safety case for ai-assisted prostate cancer diagnosis, *Computers in biology and medicine* 192 (2025) 110237.
 - [67] N. Leveson, A systems approach to risk management through leading safety indicators, *Reliability engineering & system safety* 136 (2015) 17–34.
 - [68] E. Denney, G. Pai, I. Whiteside, The role of safety architectures in aviation safety cases, *Reliability Engineering & System Safety* 191 (2019) 106502.
 - [69] G. Verma, Y. Gupta, A. M. Malik, B. Chapman, Performance evaluation of deep learning compilers for edge inference, in: *2021 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, IEEE, 2021, pp. 858–865.
 - [70] H. Wu, P. Judd, X. Zhang, M. Isaev, P. Micikevicius, Integer quantization for deep learning inference: Principles and empirical evaluation, arXiv preprint arXiv:2004.09602 (2020).
 - [71] P. Festic, Y. Jia, A. C. Gordon, A. A. Faisal, I. Habli, M. Komorowski, 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).
 - [72] M. Borg, J. Henriksson, K. Socha, O. Lennartsson, E. Sonnsjö Lönegren, T. Bui, P. Tomaszewski, S. R. Sathiamoorthy, S. Brink, M. Helali Moghadam, Ergo, SMIRK is safe: a safety case for a machine learning component in a pedestrian automatic emergency brake system, *Software Quality Journal* (2023) 1–69.
 - [73] R. Hawkins, C. Picardi, L. Donnell, M. Ireland, Creating a safety assurance case for a machine learned satellite-based wildfire detection and alert system, *Journal of Intelligent & Robotic Systems* 108 (2023) 47.
 - [74] M. A. Sujan, I. Habli, T. P. Kelly, A. Gühneemann, S. Pozzi, C. W. Johnson, How can health care organisations make and justify decisions about risk reduction? lessons from a cross-industry review and a health care stakeholder consensus development process, *Reliability Engineering & System Safety* 161 (2017) 1–11.
 - [75] ISO, PAS 21448-road vehicles-safety of the intended functionality, International Organization for Standardization (2019).
 - [76] Y. Dong, W. Huang, V. Bharti, V. Cox, A. Banks, S. Wang, X. Zhao, S. Schewe, X. Huang, Reliability assessment and safety arguments for machine learning components in system assurance, *ACM Transactions on Embedded Computing Systems* 22 (2023) 1–48.
 - [77] S. Burton, B. Herd, Addressing uncertainty in the safety assurance of machine-learning, *Frontiers in Computer Science* 5 (2023) 1132580.
 - [78] R. Bloomfield, G. Fletcher, H. Khlaaf, L. Hinde, P. Ryan, Safety case templates for autonomous systems, arXiv preprint arXiv:2102.02625 (2021).
 - [79] E. E. Alves, D. Bhatt, B. Hall, K. Driscoll, A. Murugesan, J. Rushby, Considerations in assuring safety of increasingly autonomous systems, Technical Report, NASA, 2018.
 - [80] P. Koopman, M. Wagner, Challenges in autonomous vehicle testing and validation, *SAE International Journal of Transportation Safety* 4 (2016) 15–24.
 - [81] S. Burton, L. Gauerhof, C. Heinzemann, Making the case for safety of machine learning in highly automated driving, in: *Computer Safety, Reliability, and Security: SAFECOMP 2017 Workshops, ASSURE, DECSoS, SASSUR, TELERISE, and TIPS*, Trento, Italy, September 12, 2017, Proceedings 36, Springer, 2017, pp. 5–16.
 - [82] G. Schwalbe, M. Schels, A survey on methods for the safety assurance of machine learning based systems, in: *10th European Congress on Embedded Real Time Software and Systems (ERTS 2020)*, 2020.
 - [83] R. Salay, R. Queiroz, K. Czarnecki, An analysis of ISO 26262: Using machine learning safely in automotive software, arXiv preprint arXiv:1709.02435 (2017).
 - [84] International Organization for Standardization, ISO/CD PAS 8800. Road Vehicles — Safety and artificial intelligence, under development. URL: <https://www.iso.org/standard/83303.html>.
 - [85] Y. Jia, T. Lawton, J. Burden, J. McDermid, I. Habli, Safety-driven design of machine learning for sepsis treatment, *Journal of Biomedical Informatics* 117 (2021) 103762.
 - [86] Z. Chen, Y. Bei, C. Rudin, Concept whitening for interpretable image recognition, *Nature Machine Intelligence* 2 (2020) 772–782.
 - [87] P. W. Koh, T. Nguyen, Y. S. Tang, S. Musmann, E. Pierson, B. Kim, P. Liang, Concept bottleneck models, in: *International conference on*

machine learning, PMLR, 2020, pp. 5338–5348.

- [88] C. Rudin, Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead, *Nature machine intelligence* 1 (2019) 206–215.
- [89] D. Hendrycks, M. Mazeika, S. Kadavath, D. Song, Using self-supervised learning can improve model robustness and uncertainty, *Advances in neural information processing systems* 32 (2019).
- [90] Y. Ovadia, E. Fertig, J. Ren, Z. Nado, D. Sculley, S. Nowozin, J. Dillon, B. Lakshminarayanan, J. Snoek, Can you trust your model’s uncertainty? evaluating predictive uncertainty under dataset shift, *Advances in neural information processing systems* 32 (2019).
- [91] F. Favaro, L. Fraade-Blanar, S. Schnelle, T. Victor, M. Peña, J. Engstrom, J. Scanlon, K. Kusano, D. Smith, Building a credible case for safety: Waymo’s approach for the determination of absence of unreasonable risk, *arXiv preprint arXiv:2306.01917* (2023).