Reliability Assessment and Safety Arguments for Machine Learning Components in Assuring Learning-Enabled Autonomous Systems

Xingyu Zhao[a], Wei Huang[a], Vibhav Bharti[b], Yi Dong[a], Victoria Cox[c], Alec Banks[c], Sen Wang[b], Sven Schewe[a], Xiaowei Huang (✉)[a]

[a]*Department of Computer Science, University of Liverpool, Ashton Building, Ashton Street, Liverpool, L69 3BX, U.K.*
[b]*School of Engineering & Physical Sciences, Heriot-Watt University, Edinburgh, EH14 4AS, U.K.*
[c]*Defence Science and Technology Laboratory, Salisbury, SP4 0JQ, U.K.*

Abstract

The increasing use of Machine Learning (ML) components embedded in autonomous systems – so-called Learning-Enabled Systems (LES) – has resulted in the pressing need to assure their functional safety. As for traditional functional safety, the emerging consensus within both, industry and academia, is to use assurance cases for this purpose. Typically assurance cases support claims of reliability in support of safety, and can be viewed as a structured way of organising arguments and evidence generated from safety analysis and reliability modelling activities. While such assurance activities are traditionally guided by consensus-based standards developed from vast engineering experience, LES pose new challenges in safety-critical application due to the characteristics and design of ML models. In this article, we first present an overall assurance framework for LES with an emphasis on quantitative aspects, e.g., breaking down system-level safety targets to component-level requirements and supporting claims stated in reliability metrics. We then introduce a novel model-agnostic Reliability Assessment Model (RAM) for ML classifiers that utilises the operational profile and robustness verification evidence. We discuss the model assumptions and the inherent challenges of assessing ML reliability uncovered by our RAM and propose practical solutions. Probabilistic safety arguments at the lower ML component-level are also developed based on the RAM. Finally, to evaluate and demonstrate our methods, we not only conduct experiments on synthetic/benchmark datasets but also demonstrate the scope of our methods with a comprehensive case study on Autonomous Underwater Vehicles in simulation.

Keywords: Software reliability, safety arguments, assurance cases, safe AI, robustness verification, safety-critical systems, statistical testing, operational profile, probabilistic claims, robotics and autonomous systems, autonomous underwater vehicles, safety regulation, software certification, dependable computing.

1. Introduction

Industry is increasingly adopting AI/Machine Learning (ML) algorithms to enhance the operational performance, dependability, and lifespan of products and service – systems with embedded ML-based software components. For such Learning-Enabled Systems (LES), in safety-related applications high reliability is essential to ensure successful operations and regulatory compliance. For instance, several fatalities were caused by the failures of LES built in Uber and Tesla's cars. IBM's Watson, the decision-making engine behind the Jeopardy AI success, has been deemed a costly and potentially deadly failure when extended to medical applications like cancer diagnosis. Key industrial foresight reviews have identified that the biggest obstacle to reap the benefits of ML-powered Robotics and Autonomous Systems (RAS) is the assurance and regulation of their safety and reliability [42]. Thus, there is an urgent need to

Email addresses: xingyu.zhao@liverpool.ac.uk (Xingyu Zhao), w.huang23@liverpool.ac.uk (Wei Huang), vb97@hw.ac.uk (Vibhav Bharti), yi.dong@liverpool.ac.uk (Yi Dong), vcox@dstl.gov.uk (Victoria Cox), abanks@dstl.gov.uk (Alec Banks), s.wang@hw.ac.uk (Sen Wang), sven.schewe@liverpool.ac.uk (Sven Schewe), xiaowei.huang@liverpool.ac.uk (Xiaowei Huang (✉))

develop methods that enable the dependable use of AI/ML in critical applications and, just as importantly, to *assess* and *demonstrate* the dependability for certification and regulation.

For traditional systems, safety regulation is guided by well-established standards/policies, and supported by mature development processes and Verification and Validation (V&V) tools/techniques. The situation is different for LES: they are disruptively novel and often treated as a black box with the lack of validated standards/policies [16], while they require new and advanced analysis for the complex requirements in their safe and reliable function. Such analysis needs to be tailored to fully evaluate the new character of ML [1, 19, 41], despite some progress made recently [32]. This reinforces the need for not only an overall methodology/framework in assuring the whole LES, but also innovations in safety analysis and reliability modelling for ML components, which motivate our work.

In this article, we first propose an overall assurance framework for LES, presented in Claims-Arguments-Evidence (CAE) assurance cases [14]. While inspired by [15], ours is with greater emphasis on arguing for quantitative safety requirements. This is because the unique characteristics of ML increase apparent non-determinism [36] that explicitly requires *probabilistic claims* to capture the uncertainties in its assurance [69, 3, 18]. To demonstrate the overall assurance framework as an *end-to-end* methodology, we also consider important questions on how to derive and validate (quantitative) safety requirements and how to break them down to functionalities of ML components for a given LES. Indeed, there should not be any generic, definitive, or fixed answers to those hard questions for now, since AI/ML is an emerging technology that is still heavily in flux. That said, we propose a promising solution that we believe is the most practical for the moment: we exercise the Hazard and Operability Study (HAZOP) (a systematic hazards identification method) [63], quantitative Fault-Tree Analysis (FTA) (a common probabilistic root-cause analysis) [43], and leverage existing regulation principles to validate the acceptable and tolerable safety risk, e.g., Globally At Least Equivalent (GALE) to non-AI/ML systems or human performance.

Upon establishing safety/reliability requirements on low-level functionalities of ML components, we build dedicated Reliability Assessment Models (RAM). In this article, we mainly focus on assessing the reliability of the *classification function* of the ML component, extending our initial RAM in [71] with more practical considerations for scalability. Our RAM explicitly takes the *Operational Profile (OP)* information and *robustness evidence* into account, because—(i) Reliability, as a *user-centred* property, depends on the end-users' behaviours [46], and the OP (quantifying how the software will be operated ML classifiers are subject to robustness concerns[51]) should therefore be explicitly modelled in the assessment; (ii) a RAM without considering robustness evidence is not convincing. To the best of our knowledge, our RAM is the first to consider both, the OP and robustness evidence. It is inspired by partition-based testing [28, 53], operational/statistical testing [61, 75] and ML robustness evaluation [21, 66]. Our RAM is *model-agnostic* and designed for *pre-trained* ML models, yielding estimates of, e.g., expected values or confidence bounds on the *probability of misclassification per random input (pmi)*[1].

Then, we present a set of safety case templates to support reliability claims[2] stated in *pmi* based on our new RAM—the "backbone" of the probabilistic safety arguments for ML components. Essentially, the key argument is over the rigour of the four main steps of the RAM: all perspectives of the RAM, including modelling assumptions, hyper-parameter selections, intermediate calculations and final testing results, should be presented, justified and organised in a structured way.

Finally, a comprehensive case study based on a simulated Autonomous Underwater Vehicles (AUV) that carries out survey and asset inspection missions is conducted. The case study in our simulator is both efficient and effective as a first step to demonstrate and validate our methods which, we believe, can be easily transferred to real-world case studies. All simulators, ML models, datasets and experimental results used in this work are publicly available at the our project repository https://github.com/Solitude-SAMR with a video demo at https://youtu.be/akY8f5sSFpY.

Summary of Contributions. The key contributions of this work include:

- An assurance case framework for LES that: (i) emphasises the arguments for quantitative claims on safety and reliability; (ii) with an "end-to-end" chain of safety analysis and reliability modelling methods for arguments ranging from the very top safety claim of the whole LES to low-level V&V evidence of ML components.

[1]This reliability measure is similar to the conventional *probability of failure on demand* (*pfd*), but retrofitted for classifiers.

[2]We deal with probabilistic claims in this part, so "reliability" claims are about probabilities of occurrence of failures, and "safety" claims are about failures that are safety-relevant. The two kinds do not require different statistical reasoning, thus we may use the two terms safety and reliability interchangeably when referring to the probabilities of safety-relevant failures.

- A first RAM evaluating reliability for ML software, leveraging *both* the OP information and robustness evidence. Moreover, based on the RAM, templates of probabilistic arguments for reliability claims on ML software are developed.
- Identification of open challenges in building safety arguments for LES and highlighting the inherent difficulties of assessing ML reliability, uncovered by our overall assurance framework and the proposed RAM, respectively. Potential solutions are discussed and mapped onto on-going studies to advance in this research direction.
- A prototype tool of our RAM and a simulator platform of AUV for underwater missions that are reusable and extendable as a starting point for future research.

Organisation of this Article. After presenting preliminaries in Section 2, we outline our overall assurance framework in Section 3. After that, the RAM is described in details with a running example in Section 4, following by its probabilistic safety arguments for ML classification reliability in Section 5. We then present our case study on AUV in Section 6. Related work is summarised in Section 7, while in-depth discussions are provided in Section 8. Finally, we conclude in Section 9 and outline plans for future work.

2. Preliminaries

2.1. Assurance Cases, CAE Notations and CAE Blocks

Assurance cases are developed to support claims in areas such as safety, reliability and security. They are often called by more specific names like security cases [40] and safety cases [11]. A safety case is a compelling, comprehensive, defensible, and valid justification of the system safety for a given application in a defined operating environment; it is therefore a means to provide the grounds for confidence and to assist decision making in certification [14]. For decades, safety cases have been widely used in the European safety community to assure system safety. Moreover, they are mandatory in the regulation for systems used in safety-critical industries in some countries, e.g., in the UK for nuclear energy [64]. Early research in safety cases has mainly focused on their formulation in terms of claims, arguments, and evidence elements based on fundamental argumentation theories like the Toulmin model [58]. The two most popular notations are CAE [14] and GSN [38]. In this article, we choose the former to present our assurance case templates.

A summary of the CAE notations is provided in Figure 1. The CAE safety case starts with a top *claim*, which is then supported through an *argument* by sub-claims. Sub-claims can be further decomposed until being supported by *evidence*. A claim may be subject to some context, represented by general purpose *other* nodes, while assumptions (or warranties) of arguments that need to be explicitly justified form new *side-claims*. A *sub-case* repeats a claim presented in another argument module. Notably, the basic concepts of CAE are supported by safety standards like ISO/IEC15026-2. Readers are referred to [18, 15] for more details on all CAE elements.

The CAE framework additionally consists of CAE blocks that provide five common argument fragments and a mechanism for separating inductive and deductive aspects of the argumentation[3]. These were identified by empirical analysis of real-world safety cases [17]. The five CAE blocks representing the restrictive set of arguments are:

- Decomposition: partition some aspect of the claim—"divide and conquer".
- Substitution: transform a claim about an object into a claim about an equivalent object.
- Evidence Incorporation: evidence supports the claim, with emphasis on direct support.
- Concretion: some aspect of the claim is given a more precise definition.
- Calculation (or Proof): some value of the claim can be computed or proven.

An illustrative use of CAE blocks is shown in Figure 1, while more detailed descriptions can be found in [17, 15].

[3]The argument strategy can be either inductive or deductive [1]. For an inductive strategy, additional analysis is required to ensure that residual risks are mitigated.

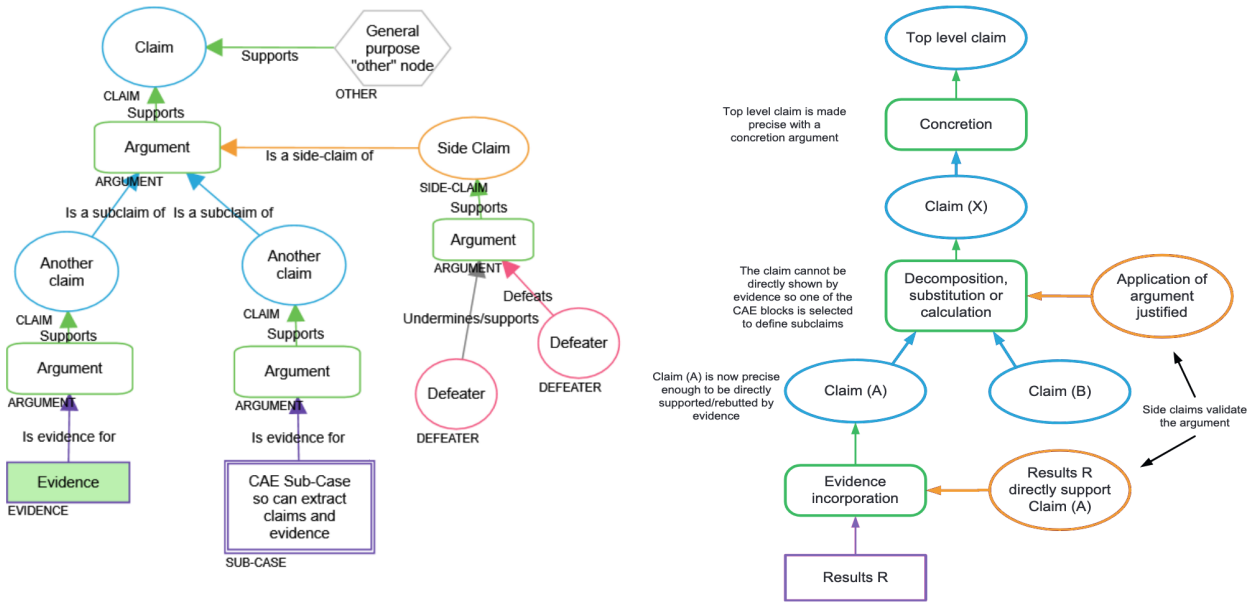

Figure 1: Summary of the CAE notations (lhs) and an example of CAE block use (rhs), cited from [15].

2.2. *HAZOP and FTA*

HAZOP is a structured and systematic safety analysis technique for risk management, which is used to identify potential hazards for the system in the given operating environment. HAZOP is based on a theory that assumes risk events are caused by deviations from design or operating intentions. Identification of such deviations is facilitated by using sets of "guide words" (e.g., too much, too little and no) as a systematic list of deviation perspectives. It is commonly performed by a multidisciplinary team of experts during brainstorming sessions. HAZOP is a technique originally developed and used in chemical industries. There are studies that successfully apply it to software-based systems [63]. Readers will see an illustrative example in later sections, while we refer to [23] for more details.

FTA is a quantitative safety analysis technique on how failures propagate through the system, i.e., how component failures lead to system failures. The fundamental concept in FTA is the distillation of system component faults that can lead to a top-level event into a structured diagram (fault tree) using logic gates (e.g., AND, OR, Exclusive-OR and Priority-AND). We show a concrete example of FTA in our case study section, while a full tutorial of developing FTA is out of the scope of this article, and readers are referred to [57] for more details.

2.3. *OP Based Software Reliability Assessment*

The *delivered reliability*, as a *user-centred* and *probabilistic* property, requires to model the end-users' behaviours (in the operating environments) and to be formally defined by a quantitative metric [46]. Without loss of generality, we focus on *pmi* as a generic metric for ML classifiers, where inputs can, e.g., be images acquired by a robot for object recognition.

Definition 1 (*pmi*)**.** *We denote the unknown pmi by a variable λ, which is formally defined as*

$$\lambda := \int_{x \in \mathcal{X}} I_{\{x \text{ causes a misclassification}\}}(x) \mathsf{Op}(x) \, \mathrm{d}x \, , \tag{1}$$

where x is an input in the input domain[4] *$\mathcal{X}$, and $I_{\mathtt{S}}(x)$ is an indicator function—it is equal to 1 when $\mathtt{S}$ is true and equal to 0 otherwise. The function $\mathsf{Op}(x)$ returns the probability that x is the next random input.*

[4]We assume continuous $\mathcal{X}$ in this article. For discrete $\mathcal{X}$, the integral in Eqn. (1) reduces to sum and $\mathsf{Op}(\cdot)$ becomes a probability mass function.

Remark 1 (OP)**.** *The OP [51] is a notion used in software engineering to quantify how the software will be operated. Mathematically, the OP is a* ***Probability Density Function (PDF)*** *defined over the whole input domain* $\mathcal{X}$*.*

We highlight this Remark 1, because we will use probability density estimators to approximate the OP from the collected operational dataset in the RAM we develop in Section 4.

By the definition of *pmi*, successive inputs are assumed to be independent. It is therefore common to use a Bernoulli process as the mathematical abstraction of the failure process, which implies a Binomial likelihood. When used for traditional software that, upon establishing the likelihood, RAMs on estimating λ vary case by case—from the basic Maximum Likelihood Estimation (MLE) to Bayesian estimators tailored for certain scenarios when, e.g., seeing no failures [50, 12], inferring ultra-high reliability [75], with certain forms of prior knowledge like perfectioness [62], with vague prior knowledge expressed in imprecise probabilities [65, 73], with uncertain OPs [13, 53], etc.

OP based RAMs designed for traditional software fail to consider new characteristics of ML, e.g., a potential lack of robustness and a high-dimensional input space. Specifically, it is quite hard to gather the required prior knowledge when taking into account the new ML characteristics in those Bayesian RAMs. At the same time, frequentist RAMs would require a large sample size to gain enough confidence in the estimates due to the extremely large population size (e.g., the high-dimensional pixel space for images). As an example, the usual accuracy testing of ML classifiers is essentially an MLE estimate against the test set, which has the following problems: (i) it assumes the test set statistically represents the OP, which is rarely the case; (ii) the test set is a very small fraction of the whole input space, thus limited confidence can be claimed in reliability; and (iii) without explicitly considering robustness evidence, the reliability claim for ML is not trustworthy.

2.4. ML Robustness and the R-Separation Property

ML is known not to be robust. Robustness requires that the decision of the ML model $\mathcal{M}$ is invariant against small perturbations on inputs. That is, all inputs in a region $\eta \subset \mathcal{X}$ have the same prediction label, where usually the region η is a small norm ball (in an L_p-norm distance[5]) of radius ϵ around an input x. Inside η, if an input x' is classified differently to x by $\mathcal{M}$, then x' is an Adversarial Example (AE). Robustness can be defined either as a binary metric (if there exists any AE in η) or as a probabilistic metric (how likely the event of seeing an AE in η is). The former aligns with formal verification, e.g. [33], while the latter is normally used in statistical approaches, e.g. [66]. The former "verification approach" is the binary version of the latter "stochastic approach"[6].

Definition 2 (robustness)**.** *Similar to [66], we adopt the more general probabilistic definition on the robustness of the model* $\mathcal{M}$ *(in a region* η *and to a target label* y*):*

$$R_{\mathcal{M}}(\eta, y) := \int_{x \in \eta} I_{\{\mathcal{M}(x) \text{ predicts label } y\}}(x) \mathsf{Op}(x \mid x \in \eta)\, \mathrm{d}x\ , \tag{2}$$

where $\mathsf{Op}(x \mid x \in \eta)$ *is the conditional OP of region* η *(precisely the "input model" used by both [66] and [67]).*

We highlight the follow two remarks regarding robustness:

Remark 2 (astuteness)**.** *Reliability assessment only concerns the robustness to the ground truth label, rather than an arbitrary label* y *in* $R_{\mathcal{M}}(\eta, y)$*. When* y *is such a ground truth, robustness becomes* ***astuteness*** *[68], which is also the* ***conditional reliability*** *in the region* η*.*

Astuteness is a special case of robustness[7]. An extreme example showing why we introduce the concept of astuteness is, that a perfectly robust classifier that always outputs "dog" for any given input is unreliable. Thus, robustness evidence cannot directly support reliability claims unless the ground truth label is used in estimating $R_{\mathcal{M}}(\eta, y)$.

Remark 3 (r-separation)**.** *For real-world image datasets, any data-points with different ground truth are at least distance* $2r$ *apart in the input space (pixel space), and* r *is bigger than a norm ball radius commonly used in robustness studies.*

[5] Distance mentioned in this article is defined in L_∞ if without further clarification.

[6] Thus, we use the more general term robustness "evaluation" rather than robustness "verification" throughout the article.

[7] Thus, later in this article, we may refer to robustness as astuteness for brevity when it is clear from the context.

The r-separation property was first observed by [68]: real-world image datasets studied by the authors implies that r is normally $3 \sim 7$ times bigger than the radius (denoted as ϵ) of norm balls commonly used in robustness studies. Intuitively it says that, although the classification boundary is highly non-linear, there is a minimal distance between two real-world objects of different classes (cf. Figure 2 for a conceptual illustration). Moreover, such a minimal distance is bigger than the usual norm ball size in robustness studies.

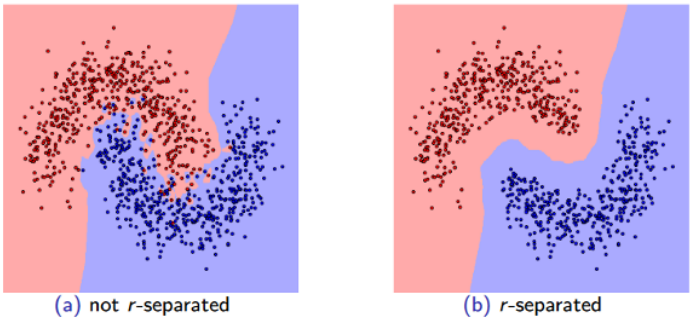

Figure 2: Illustration of the r-separation property.

3. The Overall Assurance Framework

In this section, we present an overall assurance framework for LES (e.g., AUV). The assurance framework is presented as a CAE assurance case template [14], in which we highlight both the main focus of this work—a RAM for the ML component with its probabilistic safety arguments—and all its required supporting analysis to derive the reliability requirements on the low-level ML functionalities.

3.1. Overview of an Assurance Case for LES

To argue for **TLSC1**, we refer to the template proposed by [15, Chap. 5] as our sub-case **SubC1**. Essentially, in **SubC1**, we argue *R* is: (i) well-defined (e.g., verifiable, consistent, unambiguous, traceable, etc); (ii) complete that covers all sources (e.g., from hazard analysis and domain-specific safety standards/legislation); and (iii) valid, according to some common risk acceptance criteria/principles in safety regulations of different countries/domains, e.g., ALARP (As Low As Reasonably Practicable). Without repeating the content of [15], we only highlight the parts directly supporting the main focus of this work (via the procedure in Figure 4), which are hazard identification (**SubC2**) and derivation of quantitative safety target (**SubC3**).

Similar to [15], we use a decomposition CAE-block/argument to support **TLC2**. But, in addition to time-split, we also split the claim by the qualitative and quantitative nature, since the main focus of this work, **SubC7**, concerns the probabilistic reliability modelling of ML components. Further decomposition of the whole system's quantitative requirements into functionalities of individual components (**TCL3**) is non-trivial, for which we utilise quantitative FTA. The decomposition requires a side-claim on the sufficiency of the FTA study **TLSC2**. A comprehensive development **SubC8** for **TLSC2** is out of the scope of this work, while we illustrate the gist and an example of the method in later sections. Finally, we reach the main focus of this work **SubC7** and will develop the full sub-case for it in Section 5.

3.2. Deriving Quantitative Requirements for ML Components

As mentioned, in this work we are mainly developing low-level probabilistic safety arguments, based on the dedicated RAM for ML components developed in Section 4. An inevitable question is *how to quantitatively determine the tolerable and acceptable risk levels of the ML components*. Normally the answer involves a series of well-established safety analysis methods that systematically breaks down the whole-system level risk to low-level components, considering the system architecture [45, 69]. While, the whole-system level risk is determined on a case by case basis through the application of principles and criteria required by the safety regulations extant in the different countries/domains.

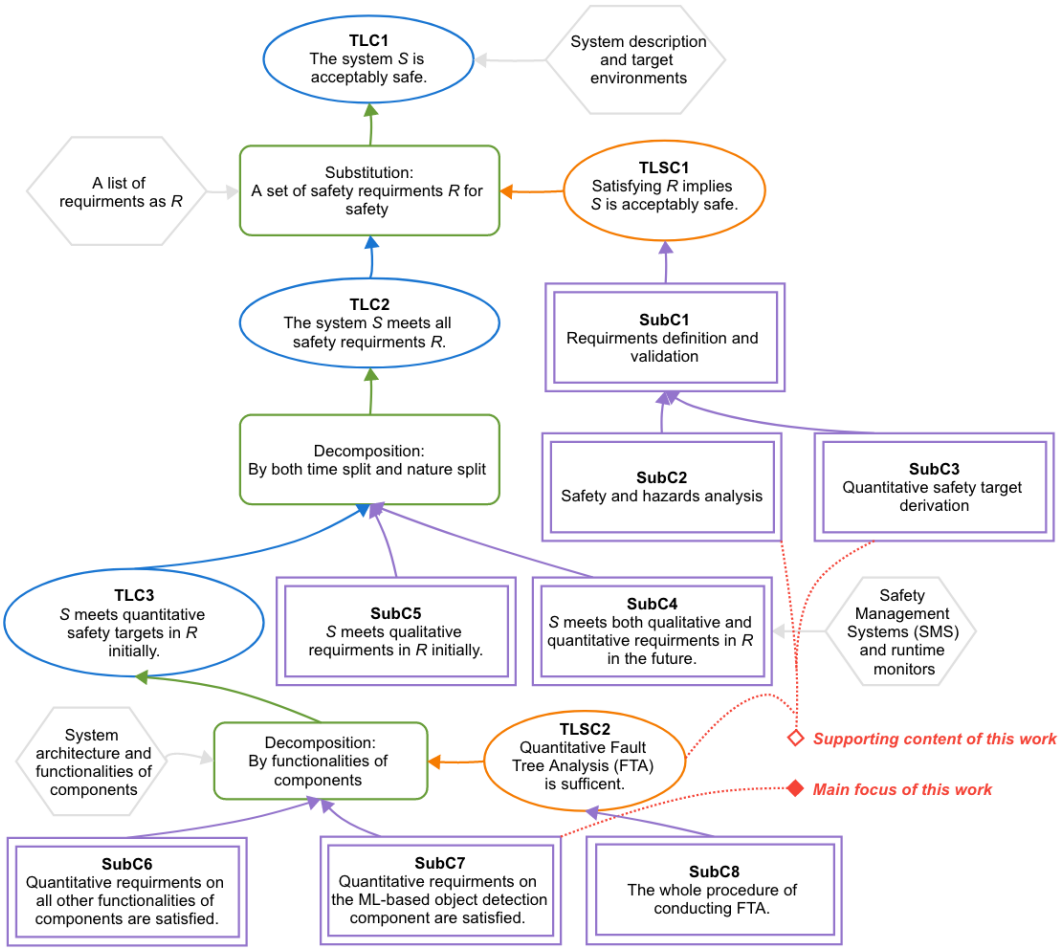

Figure 3: Overview of the proposed safety case template for LES, highlighting the main focus and supporting content of the work.

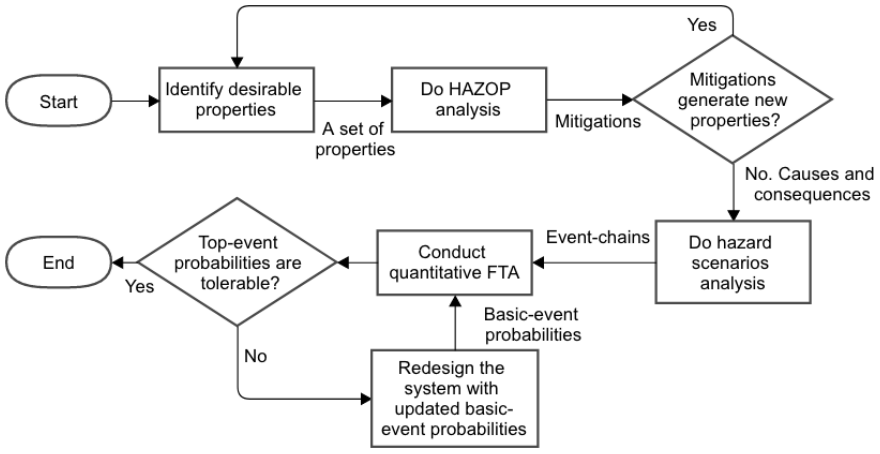

Figure 4: The workflow of combining HAZOP and quantitative FTA to derive probabilities of basic-events of components.

To align with this best practice, we propose the procedure articulated in Figure 4, whose major steps correspond to the supporting sub-cases **SubC2**, **SubC3** and **SubC8**.

In Figure 4, for the given LES, we first identify a set of safety properties that are desirable to stakeholders. Then, a HAZOP analysis is conducted, based on deviations of those properties, to systematically identify hazards (and their causes, consequences, and mitigations). New safety properties may be introduced by the mitigations identified by HAZOP, thus HAZOP is conducted in an iterative manner that forms the first loop in Figure 4.

Then, we leverage the HAZOP results to do *hazard scenario modelling*, inspired by [27], so that we may combine HAZOP and FTA later on. Usually, as also noted in [27], a property deviation can have several causes and different consequences in HAZOP analysis. It is complicated and difficult to directly convert HAZOP results into fault trees. Thus, hazard scenario modelling is needed to explicitly link the initial events (causes) to the final events (consequences) with a chain of intermediate events. Such event-chains facilitate the construction of fault trees, specifically in three steps:

- The initial events (causes) may or may not be further decomposed at even lower-level sub-functionalities of components to determine the root causes, which are used as basic events (BE) in FTA. Thus, BEs are typically failure events of software/hardware components, e.g., different types of misclassifications, failures in different modes of a propeller.
- Adding a specific logic gate among all intermediate events (IE) on the same level, which models how failures are propagated, tolerated and/or compounded throughout the system architecture.
- Final events (consequences) are used as top events (TE) of the FTA. In other words, TEs are violations of system-level safety properties.

Upon establishing the fault trees, conventional quantitative FTA can be performed to propagate probabilities of BEs to the TE probability, or, reversely, to allocate/break-down TE probability to BEs. What-if calculations and sensitivity analysis are expected to find the most practical solution of BE probabilities that makes the required TE risk tolerable. Then the practical solution for the BE associated with the ML component of our interest becomes our target reliability claims for which we develop probabilistic safety arguments. Notably, the ML component may need several rounds of retraining/fine-tuning to achieve the required level of reliability. This forms part[8] of the second iterative loop in Figure 4. We refer readers to [72] for a more detailed description on this *debug-retrain-assess* loop for ML software.

Finally, the problem boils down to (i) *how to derive the system-level quantitative safety target*, i.e., assigning probabilities for those TEs of the fault trees; and (ii) *how to demonstrate the component-level reliability is satisfied*, i.e., assessing the BE probabilities for components based on evidence. We address the second question in the next section, while the first question is essentially "how safe is safe enough?", for which the general answer depends on the *existing* regulation/certification principles/standards of different countries and industry domains. Unfortunately, existing safety standards cannot be applied on LES, and revisions are still ongoing. Therefore, we currently do not have a commonly acknowledged practice that can be easily applied to certify or regulate LES [16, 39]. That said, emerging studies on assuring/assessing the safety and reliability of AI and autonomous systems have borrowed ideas from existing regulation principles on risk acceptability and tolerability, e.g.,:

- ALARP (As Low As Reasonably Practicable): ALARP states that the residual risk after the application of safety measures should be as low as reasonably practicable. The notion of being reasonably practicable relates to the cost and level of effort to reduce risk further. It originally arises from UK legislation and is now applied in many domains like nuclear energy.
- GALE (Globally At Least Equivalent): is a principle required by French law for railway safety, which indicates the new technical system shall be at least as safe as comparable existing ones.
- SE (Substantially Equivalent): similar to GALE; new medical devices in the US must be demonstrated to be substantially equivalent to a device already on the market. This is required by the U.S. Food & Drug Administration (FDA).

[8] Other non-ML components may be updated as well to jointly make the whole-system risk tolerable.

- MEM (Minimum Endogenous Mortality): MEM states that a new system should not lead to a significant increase in the risk exposure for a population with the lowest endogenous mortality. For instance, the rate of natural deaths is a reference point for acceptability.

While a complete list of all principles and comparisons between them are beyond the scope of this work, we believe that the common trend is that, for many LES, a promising way of determining the system-level quantitative safety target is to argue the acceptable/tolerable risk over the average human-performance. For instance, self-driving cars' targets of being as safe as or two-magnitude safer than human-drivers (in terms of metrics like fatalities per mile) are studied in [37, 75, 47]. In [52], human-doctors' performance is used as the benchmark in arguing the safety of ML-based medical diagnosis systems.

In summary, we are only presenting the essential steps of combining HAZOP and quantitative FTA via *hazard scenario modelling* to derive component-level reliability requirements from whole system-level safety targets, while each of those steps with concrete examples can be found in Section 6 as part of the AUV case study.

4. Modelling the Reliability of ML Classifiers

4.1. A Running Example of a Synthetic Dataset

To better demonstrate our RAM, we take the Challenge of AI Dependability Assessment raised by Siemens Mobility[9] as a running example. The challenge is to firstly train an ML model to classify a dataset generated on the unit square $[0, 1]^2$ according to some unknown distribution (essentially the unknown OP). The collected data-points (training set) are shown in Fig. 5-lhs, in which each point is a tuple of two numbers between 0 and 1 (thus called a "2D-point"). We then need to build a RAM to claim an upper bound on the probability that the next random point is misclassified, i.e., the *pmi*. If the 2D-points represent traffic lights, then we have 2 types of misclassifications—safety-critical ones, when a red data-point is labelled green, and performance related ones otherwise. For brevity, we only focus on misclassifications here, while our RAM can cope with sub-types of misclassifications.

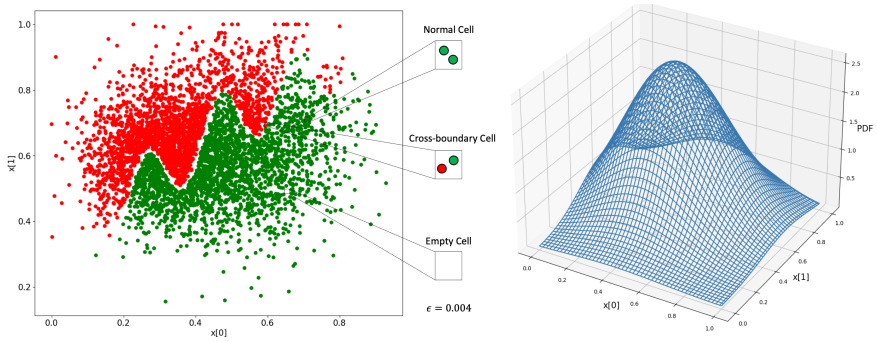

Figure 5: The 2D-point dataset (lhs), and its approximated OP (rhs).

4.2. The Proposed RAM

Principles and Main Steps of the RAM. Inspired by [53], our RAM first partitions the input domain into m small cells[10], subject to the r-separation property. Then, for each cell c_i (and its ground truth label y_i), we estimate:

$$\lambda_i := 1 - R_{\mathcal{M}}(c_i, y_i) \quad \text{and} \quad \mathsf{Op}_i := \int_{x \in c_i} \mathsf{Op}(x)\, \mathrm{d}x \,, \tag{3}$$

[9] https://ecosystem.siemens.com/topic/detail/default/33

[10] We use the term "cell" to highlight the partition that yields exhaustive and mutually exclusive regions of the input space, which is essentially a norm ball in L_∞. Thus, we use the terms "cell" and "norm ball" interchangeably in this article when the emphasis is clear from the context.

which are the *unastuteness* and *pooled OP* of the cell c_i respectively—we introduce estimators for both later. Eqn. (1) can then be written as the weighted sum of the *cell-wise* unastuteness (i.e., the conditional *pmi* of each cell[11]), where the weights are the pooled OP of the cells:

$$\lambda = \sum_{i=1}^{m} \mathsf{Op}_i \lambda_i \tag{4}$$

Eqn. (4) captures the essence of our RAM—it shows clearly how we incorporate the OP information and the robustness evidence to claim reliability. This reduces the problem is reduced to: (i) *how to obtain the estimates on those λ_is and Op_is* and (ii) *how to measure and propagate the trust in the estimates*. These two questions are challenging. To name a few of the challenges for the first question: estimating λ_i requires to determine the ground truth label of cell i; and estimating Op_is may require a large amount of operational data. For the second question, the fact that all estimators are imperfect entails that they need a measure of trust (e.g., the variance of a point estimate), which may not be easy to derive.

In what follows, by referring to the running example, we proceed in four main steps: (i) partition the input space into cells; (ii) approximate the OP of cells (the Op_is); (iii) evaluate the unastuteness of these cells (the λ_is); and (iv) "assemble" all cell-wise estimates for λ in a way that is informed by the uncertainty.

Step 1: Partition of the Input Domain $\mathcal{X}$. As per Remark 2, the astuteness evaluation of a cell requires its ground truth label. To leverage the r-separation property and Assumption 3, we partition the input space by choosing a cell radius ϵ so that $\epsilon < r$. Although we concur with Remark 3 (first observed by [68]) and believe that there should exist an *r-stable ground truth* (which means that the ground truth is stable in such a cell) for any real-world ML classification applications, it is hard to estimate such an r (denoted by $\hat{r}$) and the best we can do is to assume:

Assumption 1. *There is a r-stable ground truth (as a corollary of Remark 3) for any real-world classification problems, and the r parameter can be sufficiently estimated from the existing dataset.*

That said, in the running example, we get $\hat{r} = 0.004013$ by iteratively calculating the minimum distance of different labels. Then we choose a cell radius[12] ϵ, which is smaller than $\hat{r}$—we choose $\epsilon = 0.004$. With this value, we partition the unit square $\mathcal{X}$ into 250×250 cells.

Step 2: Cell OP Approximation. Given a dataset (X, Y), we estimate the pooled OP of cell c_i to get $\mathbb{E}[\mathsf{Op}_i]$ and $\mathbb{V}[\mathsf{Op}_i]$. We use the well-established Kernel Density Estimate (KDE) to fit a $\widehat{\mathsf{Op}}(x)$ to approximate the OP.

Assumption 2. *The existing dataset (X, Y) is randomly sampled from the OP, thus statistically represents the OP.*

This assumption may not hold in practice: training data is normally collected in a *balanced* way, since the ML model is expected to perform well in all categories of inputs, especially when the OP is unknown at the time of training and/or expected to change in future. Although our model can easily relax this assumption (cf. Section 8), we adopt it for brevity in demonstrating the running example.

Given a set of (unlabelled) data-points $(X_1, \ldots, X_n)$ from the existing dataset (X, Y), KDE then yields

$$\widehat{\mathsf{Op}}(x) = \frac{1}{nh} \sum_{j=1}^{n} K(\frac{x - X_j}{h}) \,, \tag{5}$$

where K is the kernel function (e.g. Gaussian or exponential kernels), and $h > 0$ is a smoothing parameter, called the bandwidth, cf. [60] for guidelines on tuning h. The approximated OP[13] is shown in Figure 5-rhs.

Since our cells are small and all equal size, instead of calculating $\int_{x \in c_i} \widehat{\mathsf{Op}}(x) dx$, we may approximate Op_i as

$$\widehat{\mathsf{Op}}_i = \widehat{\mathsf{Op}}\,(x_{c_i})\, v_c \tag{6}$$

[11] We use "cell unastuteness" and "cell *pmi*" interchangeably later.

[12] We use the term "radius" for cell size defined in L_∞, which happens to be the side length of the square cell of the 2D running example.

[13] In this case, the KDE uses a Gaussian kernel and $h = 0.2$ that optimised by cross-validated grid-search [8].

where $\widehat{\mathsf{Op}}(x_{c_i})$ is the probability density at the cell's central point x_{c_i}, and v_c is the constant cell volume (0.000016 in the running example).

Now if we introduce new variables $W_j = \frac{1}{h}K(\frac{x-X_j}{h})$, the KDE evaluated at x is actually the sample mean of $W_1, \ldots, W_n$. Then by invoking the Central Limiting Theorem (CLT), we have $\widehat{\mathsf{Op}}(x) \sim \mathcal{N}(\mu_W, \frac{\sigma_W^2}{n})$, where the mean is exactly the value from Eqn. (5), while the variance of $\widehat{\mathsf{Op}}(x)$ is a known result of:

$$\mathbb{V}[\widehat{\mathsf{Op}}(x)] = \frac{f(x)\int K^2(u)du}{nh} + O(\frac{1}{nh}) \approx \hat{\sigma}_B^2(x)\ , \tag{7}$$

where the last step of Eqn. (7) says that $\mathbb{V}[\widehat{\mathsf{Op}}(x)]$ can be approximated using a bootstrap variance $\hat{\sigma}_B^2(x)$ [22] (cf. Appendix A for details).

Upon establishing Eqn.s (5) and (7), together with Eqn. (6), we know for a given cell c_i (and its central point x_{c_i}):

$$\mathbb{E}[\mathsf{Op}_i] = v_c\mathbb{E}[\widehat{\mathsf{Op}}(x_{c_i})], \quad \mathbb{V}[\mathsf{Op}_i] = v_c^2\mathbb{V}[\widehat{\mathsf{Op}}(x_{c_i})]\ , \tag{8}$$

which are the OP estimates of this cell.

Step 3: Cell Astuteness Evaluation. As a corollary of Remark 3 and Assumption 1, we may confidently assume:

Assumption 3. *If the radius of c_i is smaller than r, all data-points in the cell c_i share a single ground truth label.*

Now, to determine such ground truth label of a cell c_i, we can classify our cells into three types:

- Normal cells: a normal cell contains data-points from the existing dataset. These data-points from a single cell are sharing a same ground truth label, which is then determined as the ground truth label of the cell.
- Empty cells: a cell is "empty" in the sense that it contains no data-points from the existing dataset of observed points. Some of the empty cells will eventually become non-empty as more future operational data being collected, while most of them will remain empty forever: once cells are sufficiently small, only a small share of cells will refer to physically plausible images, and even fewer are possible in a given application. For simplicity, we do not further distinguish these two types of empty cells in this paper.

  Due to the lack of data, it is hard to determine an empty cell's ground truth. For now, we do voting based on labels predicted (by the ML model) for random samples from the cell, making the following assumption.

  Assumption 4. *The accuracy of the ML model is better than a classifier doing random classifications in any given cell.*

  This assumption essentially relates to the oracle problem of ML testing, for which we believe that recent efforts (e.g. [25]) and future research may relax it.
- Cross-boundary cells: our estimate of r based on the existing dataset is normally imperfect, e.g. due to noise in the dataset, and the size of the dataset is not large enough. Thus, we may still observe data-points with different labels in a single cell (especially when new operational data with labels is collected). Such cells are crossing the classification boundary. If our estimate on r is sufficiently accurate, they should be very rare. Without the need to determine the ground truth label of a cross boundary cell, we simply and conservatively set the cell unastuteness to 1.

So far, the problem is reduced to: given a normal or empty cell c_i with the known ground truth label y_i, evaluate the misclassification probability upon a random input $x \in c_i$, $\mathbb{E}[\lambda_i]$, and its variance $\mathbb{V}[\lambda_i]$. This is essentially a statistical problem that has been studied in [66] using Multilevel Splitting Sampling, while we use the Simple Monte Carlo (SMC) method for brevity in the running example:

$$\hat{\lambda}_i = \frac{1}{n}\sum_{j=1}^{n} I_{\{M(x_j)\neq y_i\}}$$

The CLT tells us $\hat{\lambda}_i \sim \mathcal{N}(\mu, \frac{\sigma^2}{n})$ when n is large, where μ and σ^2 are the population mean and variance of $I_{\{\mathcal{M}(x_j)\neq y_i\}}$. They can be approximated with sample mean $\hat{\mu}_n$ and sample variance $\hat{\sigma}_n^2/n$, respectively. Finally, we get

$$\mathbb{E}[\lambda_i] = \hat{\mu}_n = \frac{1}{n}\sum_{j=1}^{n} I_{\{\mathcal{M}(x_j)\neq y_i\}} \tag{9}$$

$$\mathbb{V}[\lambda_i] = \frac{\hat{\sigma}_n^2}{n} = \frac{1}{(n-1)n}\sum_{j=1}^{n}(I_{\{\mathcal{M}(x_j)\neq y_i\}} - \hat{\mu}_n)^2 \tag{10}$$

Notably, to solve the above statistical problem with sampling methods, we need to assume how the inputs in the cell are distributed, i.e., a distribution for the conditional OP $\mathsf{Op}(x \mid x \in c_i)$. Without loss of generality, we assume:

Assumption 5. *The inputs in a small region like a cell are uniformly distributed.*

This assumption is not uncommon (e.g., it is made in [66, 67]) and can be easily replaced by other distributions, provided there is supporting evidence for such a change.

Step 4: Assembling of the Cell-Wise Estimates. Eqn. (4) represents an ideal case in which we know those λ_is and Op_is with certainty. In practice, we can only estimate them with imperfect estimators yielding, e.g., a point estimate with variance capturing the measure of trust[14]. To assemble the estimates of λ_is and Op_is to get the estimates on λ, and also to propagate the confidence in those estimates, we assume:

Assumption 6. *All λ_is and Op_is are independent unknown variables under estimations.*

Then, the estimate of λ and its variance are:

$$\mathbb{E}[\lambda] = \sum_{i=1}^{m}\mathbb{E}[\lambda_i \mathsf{Op}_i] = \sum_{i=1}^{m}\mathbb{E}[\lambda_i]\mathbb{E}[\mathsf{Op}_i] \text{ and} \tag{11}$$

$$\mathbb{V}[\lambda] = \sum_{i=1}^{m}\mathbb{V}[\lambda_i \mathsf{Op}_i] = \sum_{i=1}^{m}\mathbb{E}[\lambda_i]^2\mathbb{V}[\mathsf{Op}_i] + \mathbb{E}[\mathsf{Op}_i]^2\mathbb{V}[\lambda_i] + \mathbb{V}[\lambda_i]\mathbb{V}[\mathsf{Op}_i]\,. \tag{12}$$

Note that, for the variance, the covariance terms are dropped due to the independence assumption.

Depending on the specific estimators adopted, certain parametric families of the distribution of λ can be assumed, from which any quantile of interest (e.g., 95%) can be derived as our confidence bound in reliability. For the running example, we might assume $\lambda \sim \mathcal{N}(\mathbb{E}[\lambda], \mathbb{V}[\lambda])$ as an approximation by invoking the (generalised) CLT[15]. Then, an upper bound with $1-\alpha$ confidence is

$$Ub_{1-\alpha} = \mathbb{E}[\lambda] + z_{1-\alpha}\sqrt{\mathbb{V}[\lambda]}\,, \tag{13}$$

where $Pr(Z \leq z_{1-\alpha}) = 1-\alpha$, and $Z \sim \mathcal{N}(0,1)$ is a standard normal distribution.

4.3. Extension to High-Dimensional Dataset

In order to better convey the principles and main steps of our proposed RAM, we have demonstrated a "low-dimensional" version of our RAM, which is tailored for the running example (a synthetic 2D-dataset). However, real-world applications normally involve high-dimensional data like images, exposing the presented "low-dimensional" RAM to scalability challenges. In this section, we investigate how to extend our RAM for high-dimensional data, and take a few practical solutions to tackle the scalability issues raised by "the curse of dimensionality".

[14] This aligns with the traditional idea of using FTA (and hence the assurance arguments around it) for future reliability assessment.

[15] Assuming λ_is and Op_is are all normally and independently but not identically distributed, the product of two normal variables is approximately normal while the sum of normal variables is exactly normal, thus the variable λ is also approximated as being normally distributed (especially when the number of sum terms is large).

Approximating the OP in the Latent Feature Space Instead of the Input Pixel Space. The number of cells yielded by the previously discussed way of partitioning the input domain (pixel space) is exponential in the dimensionality of data. Thus, it is hard to accurately approximate the OP due to the relatively sparse data collected: the number of cells is usually significantly larger than the number of observations made. However, for real-world data (say an image), what really determines the label is its *features* rather than the pixels. Thus, we envisage some latent space, e.g. compressed by Variational Auto-Encoders (VAE), that captures only the *feature-wise* information; this latent space can be explored for high-dimensional data. That is, instead of approximating the OP in the input pixel space, we (i) first encode/project each collected data-point into the compressed latent space, reducing its dimensionality, (ii) then fit a "latent space OP" with KDE based on the compressed dataset, and (iii) finally "map" data-points (paired with the learnt OP) in the latent space back to the input space.

Remark 4 (mapping between feature and pixel spaces)**.** *Depending on which data compression technique we use and how the "decoder" works, the "map" action may vary case by case. For the VAE adopted in our work, we decode one point from the latent space as a "clean" image (with only feature-wise information), and then add perturbations to generate a norm ball (with a size determined by the r-separation distance, cf. Remark 3) in the input pixel space.*

Applying Efficient Multivariate KDE for Cell OP Approximation. We may encounter technical challenges when fitting the PDF from high-dimensional datasets. There are two known major challenges when applying *multivariate* KDE to high-dimensional data: i) the choice of bandwidth H represents the covariance matrix that mostly impacts the estimation accuracy; and ii) scalability issues in terms of storing intermediate data structure (e.g., data-points in hash-tables) and querying times made when estimating the density at a given input. For the first challenge, the optimal calculation of the bandwidth matrix can refer to some rule of thumb [60, 59] and the cross-validation [8]. There is also dedicated research on improving the efficiency of multivariate KDE, e.g., [5] presents a framework for multivariate KDE in provably sub-linear query time with linear space and linear pre-processing time to the dimensions.

Applying Efficient Estimators for Cell Robustness. We have demonstrated the use of SMC to evaluate cell robustness in our running example. It is known that SMC is not computationally efficient to estimate rare-events, such as AEs in the high-dimensional space of a robust ML model. We therefore need more advanced and efficient sampling approaches that are designed for rare-events to satisfy our need. We notice that the Adaptive Multi-level Splitting method has been retrofitted in [66] to statistically estimate the model's local robustness, which can be (and indeed has been) applied in our later experiments for image datasets. In addition to statistical approaches, formal method based verification techniques might also be applied to assess a cell's *pmi*, e.g., [33]. They provide formal guarantees on whether or not the ML model will misclassify any input inside a small region. Such "robust region" proved by formal methods is normally smaller than our cells, in which case the $\hat{\lambda}_i$ can be conservatively set as the proportion of the robust region covered in cell c_i (under Assumption 5).

Assembling a Limited Number of Cell-Wise Estimates with Informed Uncertainty. The number of cells yielded by current way of partitioning the input domain is exponential to the dimensionality of data, thus it is impossible to explore all cells for high-dimensional data as we did for the running example. We may have to limit the number of cells under robustness evaluation due to the limited budget in practice. Consequently, in the final "assembling" step of our RAM, we can only assemble a limited number of cells, say k, instead of all m cells. In this case, we refer to the estimator designed for weighted average based on samples [10]. Specifically, we proceed as what follows:

- Based on the collected dataset with n data-points, the OP is approximated in a latent space, which is compressed by VAE. Then we may obtain a set of n norm balls (paired with their OP) after mapping the compressed dataset to the input space (cf. Remark 4) as the sample frame[16].
- We define weight w_i for each of the n norm balls according to their approximated OP, $w_i := \mathbb{E}[\mathsf{Op}_i]$.
- Given a budget that we can only evaluate the robustness of k norm balls, k samples are randomly selected (with replacement) and fed into the robustness estimator to get $\mathbb{E}[\lambda_i]$.

[16]While the population is the set of (non-overlapping) norm balls covering the whole input space, i.e. the m cells mentioned in the "lower-dimensional" version of the RAM.

- We may invoke the unbiased estimator for weighted average [10, Chapter 4] as

$$\mathbb{E}[\lambda] = \frac{\sum_{i=1}^{k} w_i \mathbb{E}[\lambda_i]}{\sum_{i=1}^{k} w_i} \text{ and} \tag{14}$$

$$\mathbb{V}[\lambda] = \frac{1}{k-1}\left(\frac{\sum_{i=1}^{k} w_i \left(\mathbb{E}[\lambda_i]\right)^2}{\sum_{i=1}^{k} w_i} - \left(\mathbb{E}[\lambda]\right)^2\right) . \tag{15}$$

  Moreover, a confidence upper bound of interest can be derived from Eqn. (13).

Note that there is no variance terms of λ_i and Op_i in Eqn.s (14) and (15), implying the following assumption:

Assumption 7. *The uncertainty informed by Eqn.* (15) *is sourced from the sampling of k norm balls, which is assumed to be the major source of uncertainty. This makes the uncertainties contributed by the robustness and OP estimators (i.e. the variance terms of λ_i and Op_i) negligible.*

4.4. Evaluation on the Proposed RAM

In addition to the running example, we conduct experiments on two more synthetic 2D-datasets, as shown in Figure 6. They represent scenarios with relatively sparse and dense training data, respectively. Moreover, to gain insights on how to extend our RAM for high-dimensional datasets, we also conduct experiments on the popular MNIST and CIFAR10 datasets. Instead of implementing the steps in Section 4.2, we take solutions to tackle the scalability issues raised by "the curse of dimensionality", as articulated in Section 4.3. Finally, all modelling details and results after applying our RAM on those datasets are summarised in Table 1, where we compare the testing error, Average Cell Unastuteness (ACU) defined by Definition 3, and our RAM results (of the mean $\mathbb{E}[\lambda]$, variance $\mathbb{V}[\lambda]$ and a 97.5% confidence upper bound $Ub_{97.5\%}$).

Definition 3 (ACU)**.** *Stemmed from the Definition 2 and Remark 2, the unastuteness λ_i of a region c_i is consequently $1 - R_{\mathcal{M}}(c_i, y_i)$ where y_i is the ground truth label of c_i (cf. Eqn. 3). Then we define the ACU of the ML model as:*

$$ACU := \frac{1}{m}\sum_{i=1}^{m} \lambda_i \tag{16}$$

where m is the total number of regions.

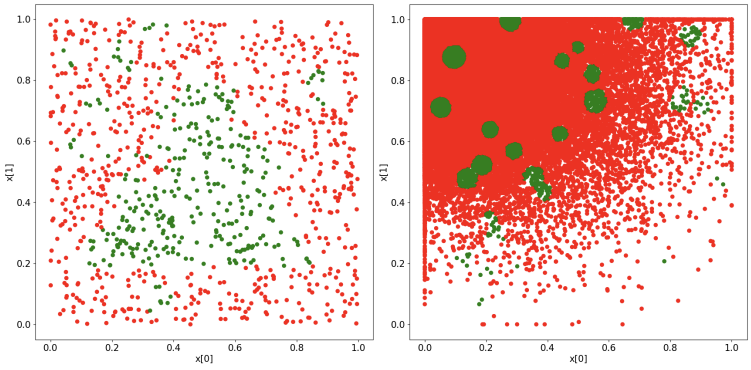

Figure 6: Synthetic datasets DS-1 (lhs) and DS-2 (rhs) representing relatively sparse and dense training data respectively.

In the running example, we first observe that the ACU is much lower than the testing error, which means that the underlying ML model is a robust one. Since our RAM is largely based on the robustness evidence, its results are close to ACU, but not exactly the same because of the nonuniform OP, cf. Figure 5-rhs.

Table 1: Modelling details and results of applying the RAM on five datasets. Time is in seconds per cell.

	train/test error	r-separation	radius ϵ	# of cells	ACU	$\mathbb{E}[\lambda]$	$\mathbb{V}[\lambda]$	$Ub_{97.5\%}$	time
The run. exp.	0.0005/0.0180	0.004013	0.004	250×250	0.002982	0.004891	0.000004	0.004899	0.04
Synth. DS-1	0.0037/0.0800	0.004392	0.004	250×250	0.008025	0.008290	0.000014	0.008319	0.03
Synth. DS-2	0.0004/0.0079	0.002001	0.002	500×500	0.004739	0.005249	0.000002	0.005252	0.04
MNIST	0.0051/0.0235	0.369	0.300	k	Fig. 7(b)	Fig. 7(a)	Fig. 7(a)	Fig. 7(a)	0.43
CIFAR10	0.0199/0.0853	0.106	0.100	k	Fig. 7(d)	Fig. 7(c)	Fig. 7(c)	Fig. 7(c)	6.74

Remark 5 (ACU is a special case of *pmi*)**.** *When the OP is "flat" (uniformly distributed), ACU and our RAM result regarding pmi are equal, which can be seen from Eqn. 4 by setting all* Op_i*s equally to* $\frac{1}{m}$.

Moreover, from Figure 5-lhs, we know that the classification boundary is near the middle of the unit square input space where misclassifications tend to happen (say, a "buggy area"), which is also the high density area on the OP. Thus, the contribution to unreliability from the "buggy area" is weighted higher by the OP, explaining why our RAM results are worse than the ACU. In contrast, because of the relatively "flat" OP for the DS-1 (cf. Figure 6-lhs), our RAM result is very close to the ACU (cf. Remark 5). With more dense data in DS-2, the r-distance is much smaller and leads to smaller cell radius and more cells. Thanks to the rich data in this case, all three results (testing error, ACU, and the RAM) are more consistent than in the other two cases. We note that, given the nature of the three 2D-point datasets, ML models trained on them are much more robust than image datasets. This is why all ACUs are better than test errors, and our RAM finds a middle point representing reliability according to the OP. Later we apply the RAM on two unrobust ML models trained on image datasets, where the ACUs are worse than the test error; it confirms our aforementioned observations.

Regarding the MNIST and CIFAR10 datasets, we first train VAE on them and compress the datasets into the low dimensional latent spaces of VAE with 8 and 16 dimensions, respectively. We then fit the compressed dataset with KDE to approximate the OP. Each compressed data-point is now associated with a weight representing its OP. Consequently, each norm ball in the pixel space that corresponds to the compressed data-point in the latent space (after the mapping, cf. Remark 4) is also weighted by the OP. Taking the computational cost into account—say only the astuteness evaluation on a limited number of k norm balls is affordable—we do random sampling, invoke the estimator for *weighted average* Eqn.s (14) and (15), and plot our RAM results as functions of k in Figure 7(a) and 7(c). For comparison, we also plot the ACU results[17] in Figure 7(b) and 7(d).

In Figure 7, we first observe that both, the ACU results (after converging) of MNIST and CIFAR10, are worse than their test errors (in Table 1), unveiling again the robustness issues of ML models when dealing with image datasets (while the ACU of CIFAR10 is even worse, given that CIFAR10 is indeed a generally harder dataset than MNIST). For MNIST, the mean *pmi* estimates are much lower than ACU, implying a very "unbalanced" distribution of weights (i.e. OP). Such unevenly distributed weights are also reflected in both, the oscillation of the variance and the relatively loose 97.5% confidence upper bound. On the other hand, the OP of CIFAR10 is flatter, resulting in closer estimates of *pmi* and ACU (Remark 5). In summary, for real-world image datasets, our RAM may effectively assess the robustness of the ML model and its generalisability based on the shape of its approximated OP, which is much more informative than either the test error or ACU alone.

5. Probabilistic Safety Arguments for ML Components

At this lower level of ML components, cf. the **SubC7** in Figure 3, we further decompose and organise our safety arguments in two levels—*decomposing sub-functionalities of ML components doing object detection* and *claiming the reliability of the classification function*. In the following sections, we discuss both of them in details, while focusing more on the latter.

[17] As per Remark 5, ACU is a special case of *pmi* with equal weights. Thus, ACU results in Figure 7 are also obtained by Eqn.s (14) and (15).

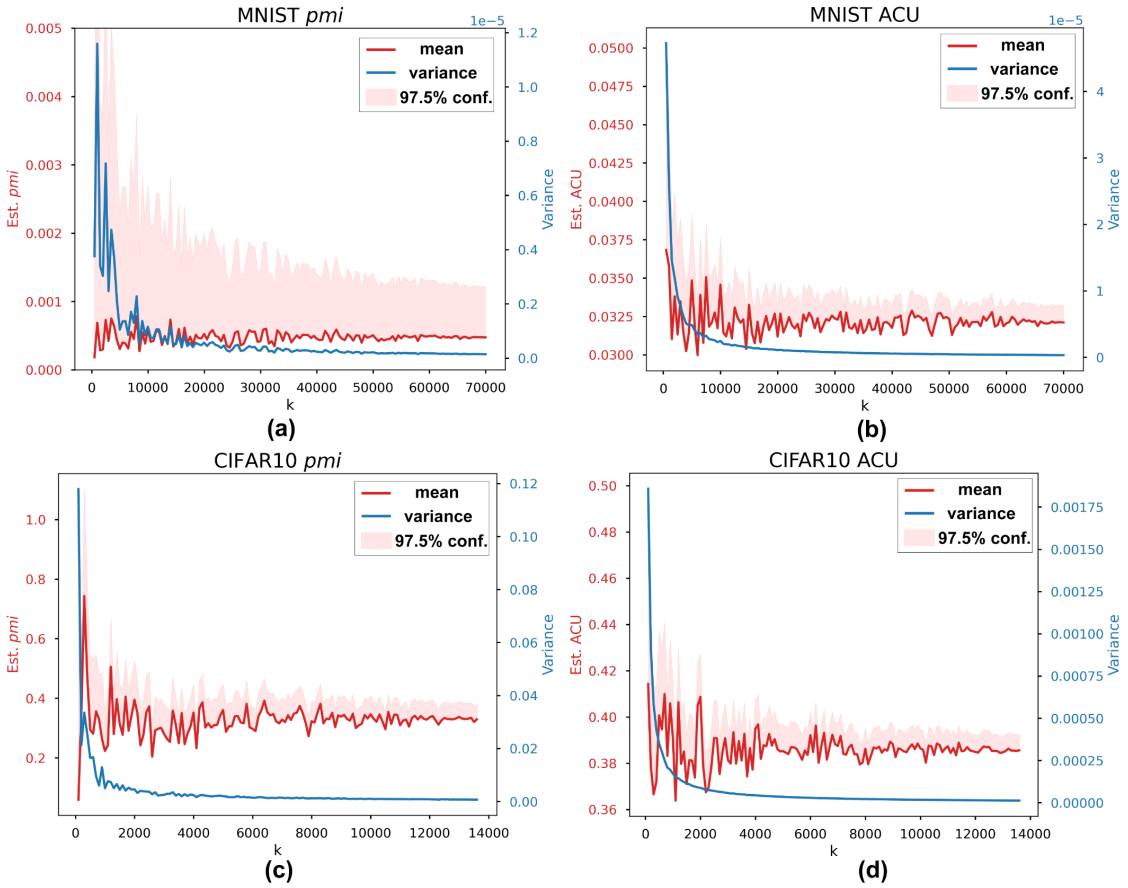

Figure 7: The mean, variance and 97.5% confidence upper bound of *pmi* and ACU as functions of k sampled norm balls.

5.1. Arguments for Top Claims on Object Detection at the ML Component-Level

In Figure 8, we present an argument template, again in the CAE blocks at the ML component-level. It aims at breaking down the claim "The object detection is safe enough" **LLC1** to a reliability claim stated in the specified measure. The first argument is over all safety related properties, and presented by a CAE block of substitution. The list of all properties of interest for the given application can be obtained by utilising the Property Based Requirements (PBR) approach [49], forming the side-claim **LLSC1**, which is supported by the sub-case **SubC10**. The PBR analysis, recommended in [1] as a method for safety arguments of autonomous systems, is a way to specify requirements as a set of properties of system objects either in a structured language or formal notations. In this work, we focus on the main quantitative property—reliability—while other properties like security and interpretability are omitted and remain an undeveloped sub-case **SubC9** in the CAE template.

Starting from **LLC2**, we then argue over the decomposition by four sub-functionalities of object detection. At the "birth" of an object in the system's vision (e.g., the total number of pixels is greater than a threshold), the ML component should accurately classify it, localise it (normally measured by the Intersection over Union (IoU) of bounding boxes) and in a good timing (e.g., no later than some frames after its birth). Once initially detected at its birth time, the tracking function on that object should be reliable enough to make decision making by other control components safe. The four sub-functionalities of object detection forms the claims **LLC3**-**LLC5**.

To support the reliability of classification at the birth time of the objects **LLC3**, we concretise the reliability requirements in terms of specific reliability measures, in our case *pmi*. The "misclassification" and "per input" in *pmi* need to be clearly defined: (i) we only consider safety-related misclassification events; (ii) an input refers to the image frame capturing the "birth" of an object in the camera's vision (so that images can be treated as independent

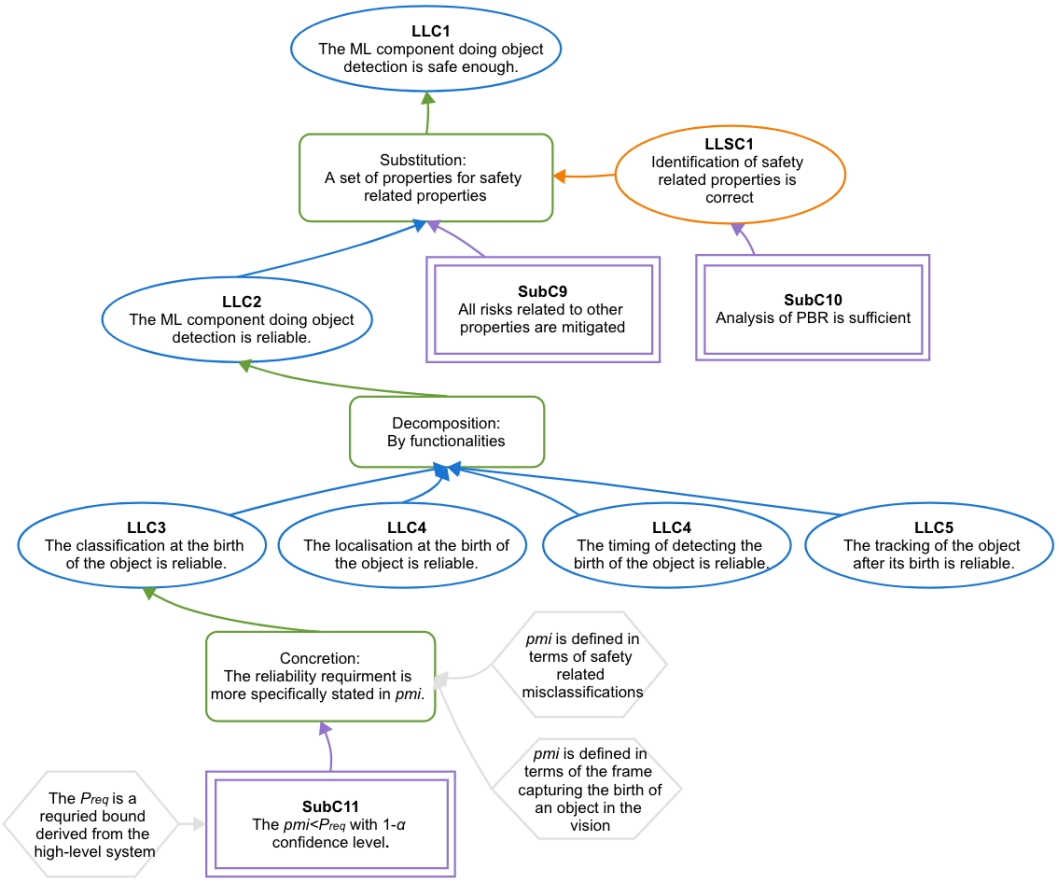

Figure 8: ML component-level arguments breaking down the claim "The object detection component is safe enough" **LLC1** to reliability claims of the classification function stated in specific reliability measures **SubC11**.

conforming to the definition of *pmi*). We are then interested in the claim of a bound on *pmi* with $(1-\alpha)$ confidence, where P_{req} is a required bound derived from higher level safety analysis.

While the reliability of the other three sub-functionalities can be similarly concretised by some quantitative measures, e.g. IoU for localisation, they remain undeveloped in this article and form important future work.

5.2. Low-Level Arguments for Classification Based on the RAM

In this section, we present **SubC11** and show how to support a reliability claim stated in *pmi* based on our RAM developed in Section 4—the "backbone" of the probabilistic arguments at this lower level. Essentially, we argue over the four main steps of our RAM as shown in Figure 9. Note that, depending on the data dimensionality of the specific application, we may either use the "low-dimensional" version of our RAM, where the whole input space is partitioned into cells, or apply the "high-dimensional" version, in which norm balls (of relatively spare data) are determined instead (cf. Remark 4) based on the collected data to form the sample frame (representing the population of all norm balls partitioning the whole input space). Indeed, the method of exhaustively partitioning cells is also applicable to high-dimensional data, but it would yield an extremely large number of cells that is not only infeasible to exhaustively examine them but also quite difficult to index for sampling. That said, for high-dimensional datasets, we determine norm balls from the data instead, forming a smaller and more practical sampling frame. However, the price paid is at introducing two more noise factors in the assurance—the bias/error from the construction of the sampling frame and the relatively low sample rate. The former can be mitigated by conventional ways of checking (and rebuilding if necessary) the sampling frame, while the latter has been captured and quantified by the variance of the point estimate (cf. Eqn. (15) and Assumption 7).

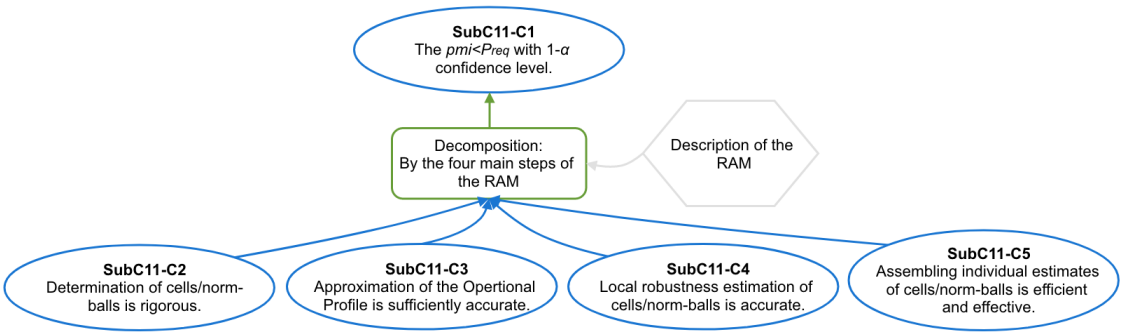

Figure 9: Arguments over the four main steps in the proposed RAM.

Figures 10 to 13 show the arguments based on steps 1 to 4 of our RAM, respectively. While the arguments presented in CAE are self-explanatory together with the technical details articulated in Section 4, we note that i) all modelling assumptions are presented as side-claims of arguments that need more application-specific development and justification; and ii) the development of some claims are omitted for brevity, because they are generic claims and thus can be referred to other works, e.g. [4], for **SubC11-C3.2** and **SubC11-C3.3** when we treat the OP estimator as a common data-driven learning model.

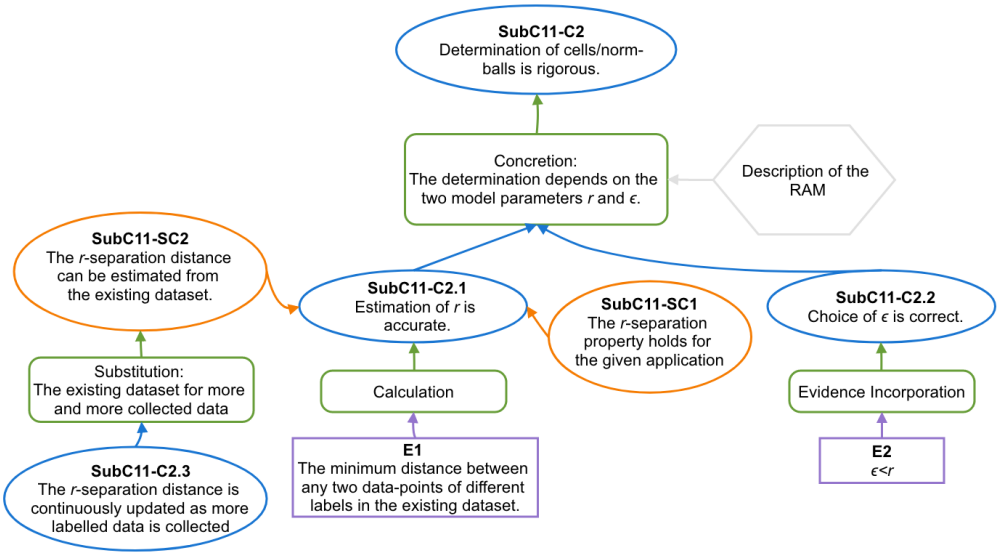

Figure 10: Arguments based on the step 1 of the RAM.

6. A Case Study of AUV Missions

In this section, a case study based on a simulated AUV that performs survey and asset inspection missions is conducted. We first describe the scenario in which the mission is performed, details of the AUV under test, and how the simulator is implemented. Then, corresponding to Section 3, we exercise the proposed assurance activities for this AUV application, i.e., HAZOP, hazards scenarios modelling, FTA, and discussions on deriving the system-level quantitative safety target for this scenario. Finally, we apply our RAM on the image dataset collected from a large amount of statistical testing. All source code, simulators, ML models, datasets and experiment results are publicly available on our project website https://github.com/Solitude-SAMR/master_samr with a video demo at https://youtu.be/akY8f5sSFpY.

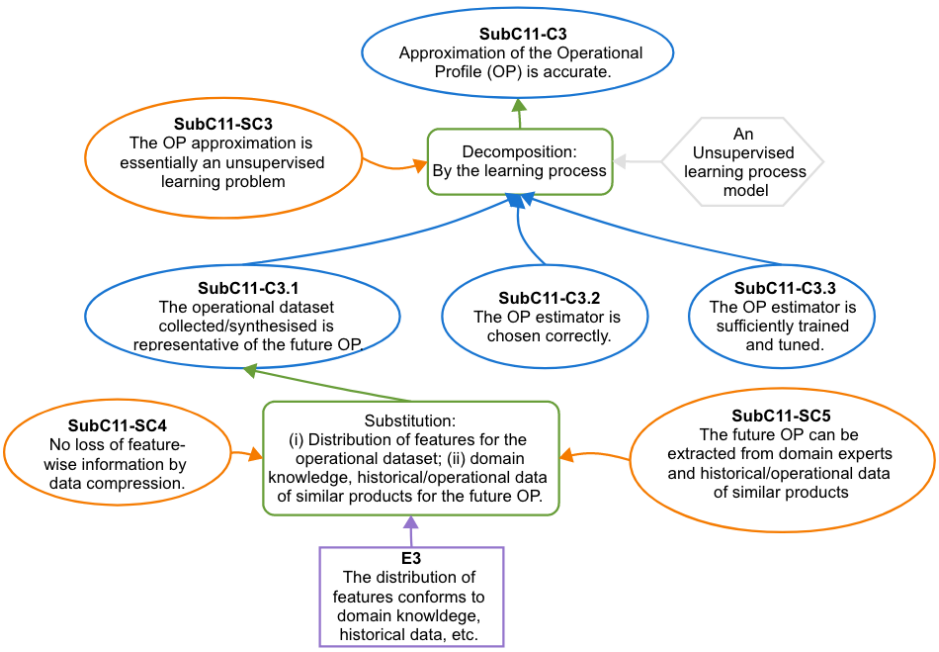

Figure 11: Arguments based on the step 2 of the RAM.

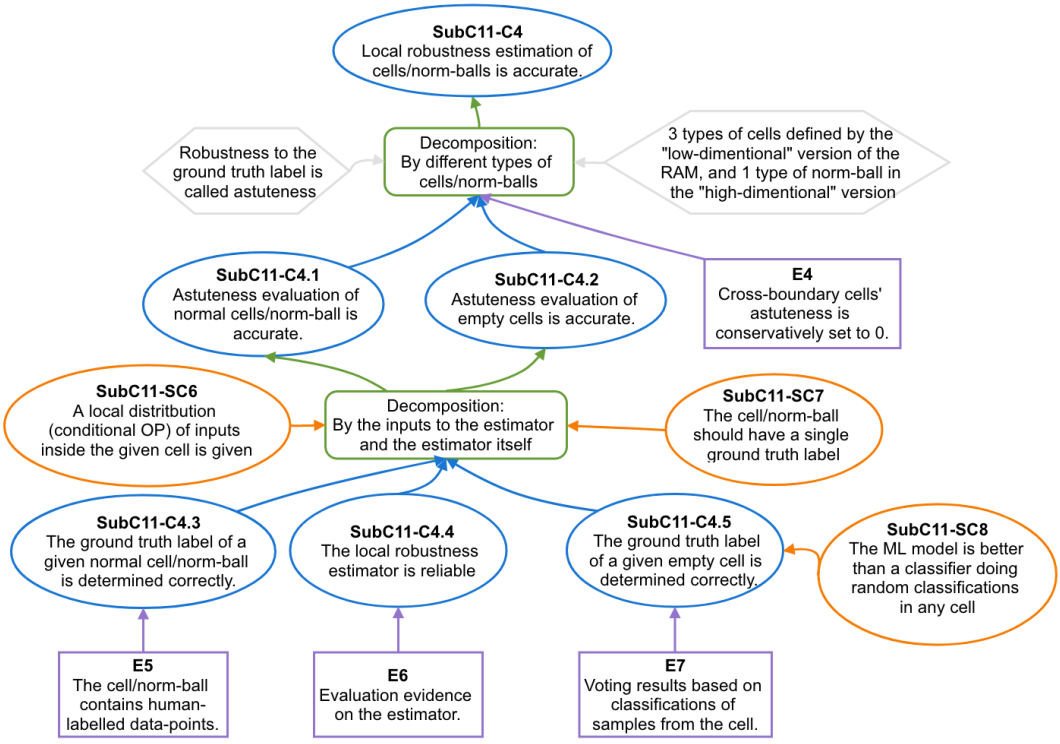

Figure 12: Arguments based on the step 3 of the RAM.

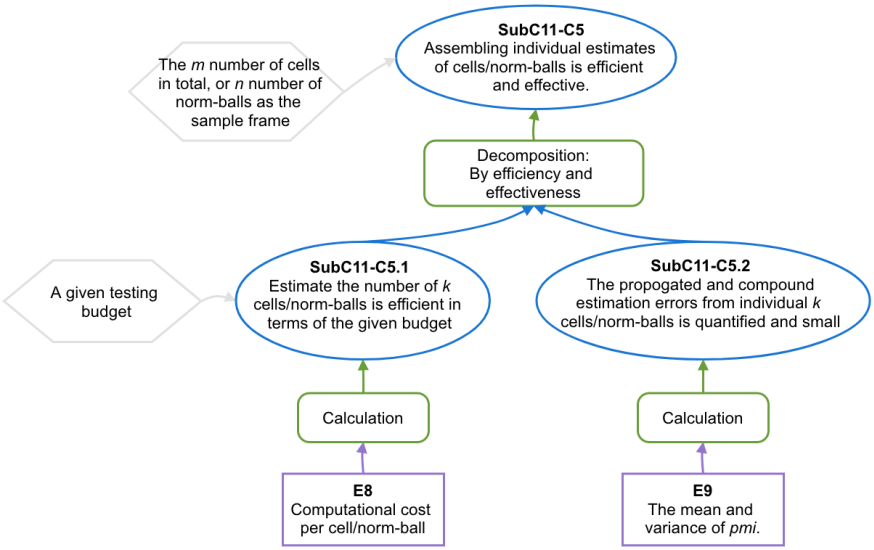

Figure 13: Arguments based on the step 4 of the RAM.

6.1. Scenario Design

AUV are increasingly adopted for marine science, offshore energy, and other industrial applications in order to increase productivity and effectiveness as well as to reduce human risks and offshore operation of crewed surface support vessels [42]. However, the fact that AUVs frequently operate in close proximity to safety-critical assets (e.g., offshore oil rigs and wind turbines) for inspection, repair and maintenance tasks leads to challenges on the assurance of their reliability and safety, which motivates the choice of AUV as the object of our case study.

6.1.1. Mission Description and Identification of Mission Properties

Based on industrial use cases of autonomous underwater inspection, we define a test scenario for AUVs that need to operate autonomously and carry out a survey and asset inspection mission, in which an AUV follows several way-points and terminates with autonomous docking. During the mission, it needs to detect and recognise a set of underwater objects (such as oil pipelines and wind farm power cables) and inspect assets (i.e., objects) of interest, while avoiding obstacles and keeping the required safe distances to the assets.

Given the safety/business-critical mission, different stakeholders have their own interests on a specific set of hazards and safety elements. For instance, asset owners (e.g., wind farm operators) focus more on the safety and health of the assets that are scheduled to be inspected, whereas inspection service providers tend to have additional concerns regarding the safety and reliability of their inspection service and vehicles. In contrast, regulators and policy makers may be more interested in environmental and societal impacts that may arise when a failure unfortunately happens. By keeping these different safety concerns in mind, we identify a set of desirable **mission properties**, whose violation may lead to unsuccessful inspection missions, compromise the integrity of critical assets, or damage of the vehicle itself.

While numerous high-level mission properties are identified based on our engineering experience, references to publications (e.g., [30]) and iterations of hazard analysis, we focus on a few that are instructive for the ML classification function in this article (cf. the project website for a complete list):

- No miss of key assets: the total number of correctly recognised assets/objects should be equal to the total number of assets that are required to be inspected during the mission.
- No collision: during the full mission, the AUV should avoid all obstacles perceived without collision.
- Safe distancing: once an asset is detected and recognised, the Euclidean distance between the AUV and the asset must be kept to be at least the defined minimal safe operating distance.

- Autonomous docking: safe and reliable docking to the docking cage.

Notably, such an initial set of desirable mission properties forms the starting point of our assurance activities, cf. Figure 4 and Section 6.2.

6.1.2. The AUV Under Test

Hardware. Although we are only conducting experiments in simulators at this stage, our trained ML model can be easily deployed to real robots and the experiments are expected to be reproducible in real water tanks. Thus, we simulate the AUV in our laboratory—a customised BlueROV2, which has 4 vertical and 4 horizontal thrusters for 6 degrees of freedom motion. As shown in Figure 14-lhs, it is equipped with a custom underwater stereo camera designed for underwater inspection. A Water Linked A50 Doppler Velocity Log (DVL) is installed for velocity estimation and control. The AUV also carries an Inertial Measurement Unit (IMU), a depth sensor and a Tritech Micron sonar. The AUV is extended with an on-board Nvidia Jetson Xavier GPU computer and a Raspberry Pi 4 embedded computer. An external PC can also be used for data communication, remote control, mission monitoring, and data visualisation of the AUV via its tether.

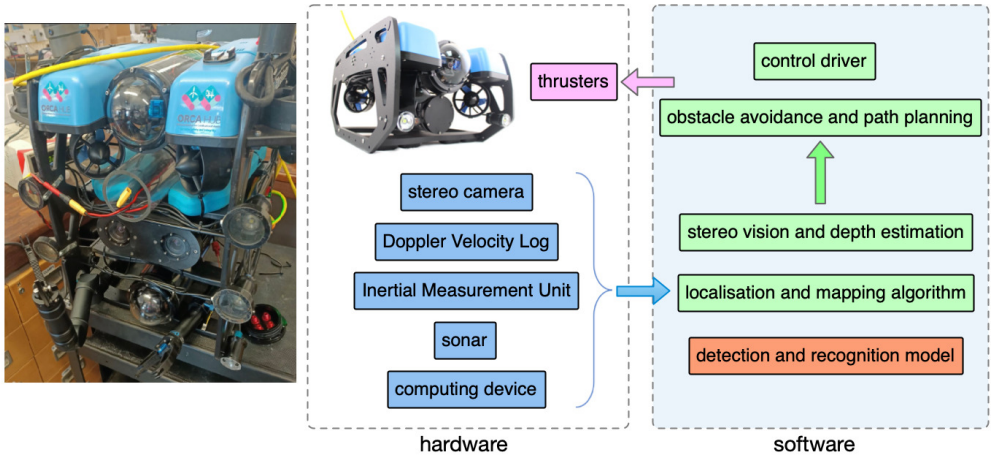

Figure 14: Hardware and software architecture and key modules for autonomous survey and inspection missions.

Software Architecture. With the hardware platform, we develop a software stack for underwater autonomy based on the Robot Operating System (ROS). The software modules that are relevant to the aforementioned AUV missions are (cf. Figure 14):

- Sensor drivers. All sensors are connected to on-board computers via cables, and their software drivers are deployed to capture real-time sensing data.
- Stereo vision and depth estimation. This is to process stereo images by removing its distortion and enhancing its image quality for inspection. After rectifying stereo images, they are used for estimating depth maps that are used for 3D mapping and obstacle avoidance.
- Localisation and mapping algorithm. In order to navigate autonomously and carry out a mission, we need to localise the vehicle and build a map for navigation. We develop a graph optimisation based underwater simultaneous localisation and mapping system by fusing stereo vision, DVL, and IMU. It also builds a dense 3D reconstruction model of structures for geometric inspection.
- Detection and recognition model. This is one of the core modules for underwater inspection based on ML models. It is designed to detect and recognise objects of interest in real-time. Based on the properties of detected objects— in particular the underwater assets to inspect—the AUV makes decisions on visual data collection and inspection.

- Obstacle avoidance and path planning. The built 3D map and its depth estimation are used for path planning, considering obstacles perceived by the stereo vision. Specifically, a local trajectory path and its way-points are generated in the 3D operating space based on the 3D map built from the localisation and mapping algorithm. Next the computed way-point is passed to the control driver for trajectory and way-point following.

- Control driver. We have a back seat driver for autonomous operations, enabling the robot to operate as an AUV. Once the planned path and/or a way-point is received, a proportional–integral–derivative (PID) based controller is used to drive the thrusters following the path and approaching to the way-point. The controller can also be replaced by a learning based adaptive controller. While the robot moves in the environment, it continues perceiving the surrounding scene and processing the data using the previous software modules.

ML Model Doing Object Detection. In this work, the state-of-the-art Yolo-v3 Deep Learning (DL) architecture [55] is used for object detection. Its computational efficiency and real-time performance are both critical for its application for underwater robots, as they mostly have limited on-board computing resources and power. The inference of Yolo can be up to 100 frames per second. Yolo models are also open source and built using the C language and the library is officially supported by OpenCV, which makes its integration with other AUV systems not covered in this work straightforward. Most DL-based object detection methods are extensions of a simple classification network. The object detection network usually generates a set of proposal bounding boxes; they might contain an object of interest and are then fed to a classification network. The Yolov3 network is similar in operation to, and is based on, the *darknet53* classification network.

The process of training the Yolo networks using the Darknet framework is similar to the training of most ML models, which includes data collection, model architecture implementation, and training. The framework consists of configuration files that can be set to match the number of object classes and other network parameters. Examples of training and testing data are described in Section 6.1.3 for simulated version of the model. The model training can be summarised by the following steps: i) define the number of object categories; ii) collect sufficient data samples for each category; iii) split the data into training and validation sets; and iv) use the Darknet software framework to train the model.

6.1.3. The Simulator

The simulator uses the popular Gazebo robotics simulator in combination with a simulator for underwater dynamics. The scenario models can be created/edited using Blender 3D software. We have designed the Ocean Systems Lab's wave tank model (cf. Figure 15-lhs) for the indoor simulated demo, using BlueROV2 within the simulation to test the scenarios. The wave tank model has the same dimension as our real tank. To ensure that the model does not

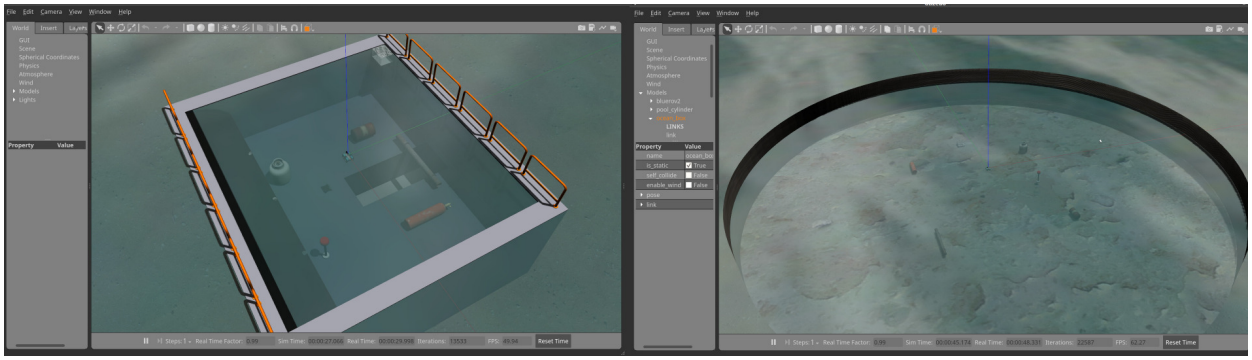

Figure 15: A wave-tank for simulated testing and a simulated pool for collecting the training data.

overfit the data, we have designed another scenario with a bigger pool for collecting the training data. The larger size allows for more distance between multiple objects, allowing both to broaden the set training scenarios and to make them more realistic. The simulated training environment is presented in Figure 15-rhs.

Our simulator creates configuration files to define an automated path using Cartesian way-points for the vehicle to follow autonomously, which can be visualised using Rviz. The pink trajectory is the desirable path and the red

arrows represent the vehicle poses following the path, cf. Figure 16-lhs. There are six simulated objects in the water tank. They are a pipe, a gas tank, a gas canister, an oil barrel, a floating ball, and the docking cage, as shown in Figure 16-rhs. The underwater vehicle needs to accurately and timely detect them during the mission. Notably, the mission is also subject to random noise factors, so that repeated missions will generate different data that is processed by the learning-enabled components.

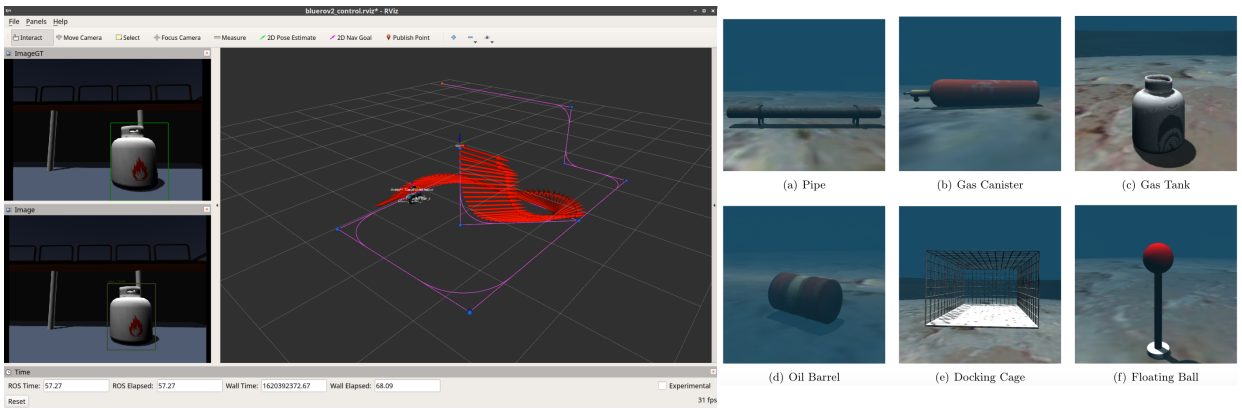

Figure 16: Simulated AUV missions following way-points and the six simulated objects.

6.2. Assurance Activities for the AUV

Hazard Analysis via HAZOP. Given the AUV system architecture (cf. Figure 14) and control/data flow among the nodes, we may conduct a HAZOP analysis that yields a complete version of Table 2. For this work, we only present partial HAZOP results and highlight a few hazards that are due to misclassification.

HAZOP item: node/flow	Process parameter or attribute	Guide-word	Cause	Consequence	Mitigation
flow from object detection to obstacle avoidance and path planning	data flow	too late	...	...	...
		...	...	...	...
	data value	wrong value	misclassification	erratic navigation; unsafe distance to assets; collision to assets; failed inspection.	acoustic guidance; minimum DL-classifier reliability for critical objects; maximum safe distance maintained if uncertain; ...
		no value	...	...	...
		...	...	...	...
...	...	...	...	...	...

Table 2: Partial HAZOP results, highlighting the cause of misclassification (NB, entries of "..." are intentionally left blank).

Hazard Scenarios Modelling. Inspired by [27], we have develop the hazard scenarios as chains of events that link the causes to consequences identified by HAZOP. Again, for illustration, a single event-chain is shown in Figure 17, which propagates the event of misclassification on assets via the system architecture to the violation of mission property of keeping a safe distance to assets. Later, readers will see this event-chain forms one path of a fault tree in the FTA in Figure 18.

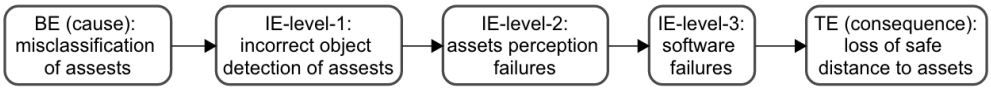

Figure 17: A single event-chain based on the hazard scenario modelling, linking causes to consequences.

Quantitative FTA. We first construct fault trees for each hazard (as TE) identified by HAZOP, by extending and combining (via logic gates) the IEs modelled by hazard scenario analysis. Each event-chain yielded by the hazard scenario analysis then forms one path in a fault tree. For instance, the event-chain of Figure 17 eventually becomes the path of **BE-0-1** $\rightarrow$ **IE-1-1** $\rightarrow$ **IE-2-2** $\rightarrow$ **IE-3-2** $\rightarrow$ **TE** in Figure 18. Finally, knowing the probabilities of BEs and logic gates allows for the calculation of the TE probability. As shown by the second iteration loop in Figure 4, several rounds of what-if calculations, sensitivity analysis and updates of the components are expected to yield the most practical solution of BE probabilities that associates with a given tolerable risk of the TE.

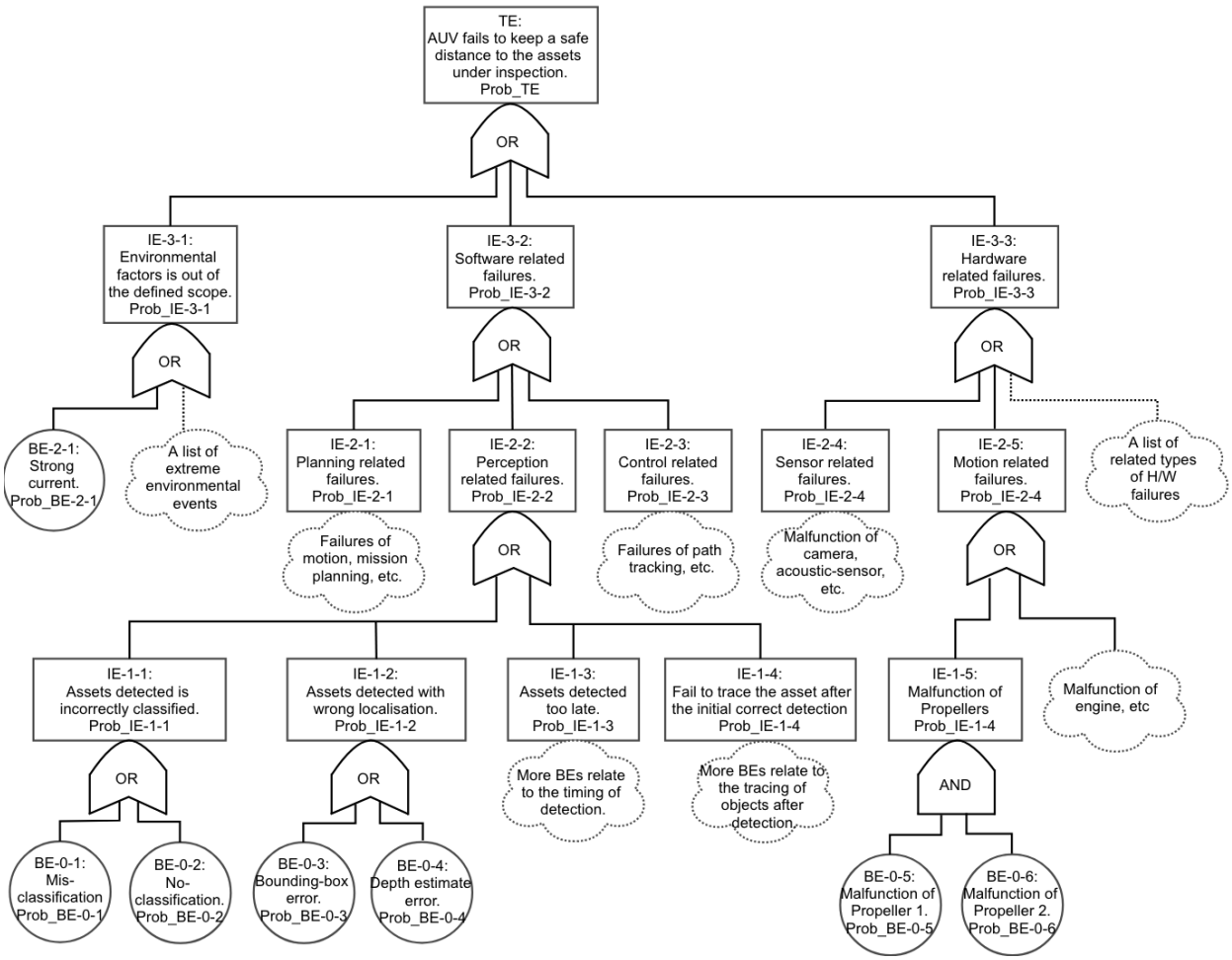

Figure 18: A partial fault tree for the TE of loss of a safe distance to assets. NB, the "cloud" notation represents omitted sub-trees.

Deriving Quantitative System Safety Target. Based on the experience of relatively more developed safety-critical domains of AI, such as self-driving cars and medical devices (cf. Section 3.2 for some examples), we believe that referring to the average performance of human divers and/or human remote control operators is a promising way of determining the high-level quantitative safety target for our case of an AUV. It is presumed that, prior to the use of an

AUV for assets inspection, human divers and remotely controlled robots need to conduct the task regularly. This is also similar to how the safety targets were developed in the civil aircraft sector where they refer to acceptable historical accident rates as the benchmark. In our case, referring to the human-divers/operators' performance as a target for an AUV's safety risk can be potentially impeded by the lack of historical/statistical data on such performance. Given the fact that ML model for AUV is a relatively novel technique and still developing and transforming to its practical uses, an urgent lesson learnt for all AUV stakeholders (especially manufacturers, operators and end users) from this work is to collect and summarise such data.

6.3. Reliability Modelling of the AUV's Classification Function

Details of the Yolo3 model trained in this case study is presented in Table B.3, Appendix B. We adopt the practical solutions discussed in Section 4.3 to deal with the high dimensionality of the collected operational dataset (256*256*3) by first training a VAE model and compressing the dataset into a new space with a much lower dimensionality of 8. While training details of the VAE model are summarised in Table B.4, four sets of examples are shown in Figure B.20, from which we can see that the reconstructed images are preserving the essential features of the objects (while blurring the less important background). We then choose a norm ball radius $\epsilon = 0.06$ according to the r-separation distance[18] and invoke the KDE and robustness estimator [66] for k randomly selected norm balls. Individual estimates of the k norm balls are then fed into the estimator for weighted average, Eqn.s (14) and (15). For comparison, we also calculate the ACU by assuming equal weights (i.e., a flat OP) in Eqn.s (14) and (15). Finally, the reliability claims on *pmi* and ACU are plotted as functions of k in Figure 19. Interpretation of the results is similar as before for CIFAR10, where the OP is also relatively flat.

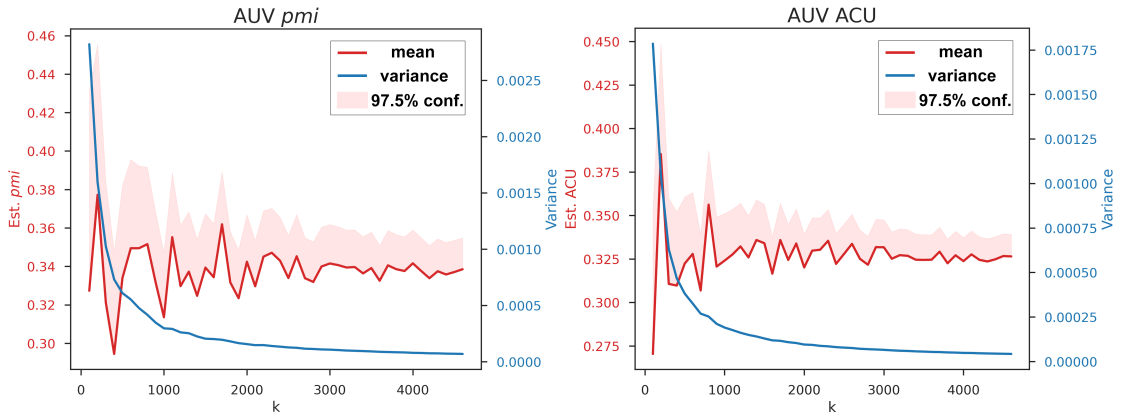

Figure 19: The mean, variance and 97.5% confidence upper bound of AUV's *pmi* and ACU as functions of k sampled norm balls.

7. Related Work

Assurance Cases for AI/ML-powered Autonomous Systems. Work on safety arguments and assurance cases for AI/ML models and autonomous systems has emerged in recent years. Burton *et al.* [19] draw a broad picture of assuring AI and identify/categorise the "gap" that arises across the development process. Alves *et al.* [1] present a comprehensive discussion on the aspects that need to be considered when developing a safety case for increasingly autonomous systems that contain ML components. Similarly, in [16], an initial safety case framework is proposed with discussions on specific challenges for ML, which is later implemented with more details in [15]. A recent work [35] also explicitly suggests the combination of HAZOP and FTA in safety cases for identifying/mitigating hazards and deriving safety

[18] Because more than one object may appear in a single image, the label of the "dominating" object (e.g., the object with the largest bounding box and/or with higher priority) can be used in the calculation of r. For simplicity, we first preprocess the dataset by filtering out images with multiple labels, and then determine the ϵ based on an estimated r.

requirements (and safety contracts) when studying Industry 4.0 systems. In [41], safety arguments that are being widely used for conventional systems—including conformance to standards, proven in use, field testing, simulation, and formal proofs—are recapped for autonomous systems with discussions on the potential pitfalls. Both, [48] and [34], propose utilising continuously updated arguments to monitor the weak points and the effectiveness of their countermeasures, while a similar mechanism is also suggested in our assurance case, e.g., continuously monitor/estimate key parameters of our RAM—all essentially aligns with the idea of dynamic assurance cases [20, 2].

Although the aforementioned works have inspired this article, our assurance framework is with greater emphasis on, and thus complements them from, the quantitative aspects, e.g., reasoning for reliability claims stated in bespoke measures and breaking down system-level safety targets to component-level quantitative requirements. Also exploring quantitative assurance, Asaadi *et al.* [3] identifies dedicated assurance measures that are tailored for properties of aviation systems.

OP-based Software Testing. OP-based software testing, also known as statistical/operational testing [61], is an established practice, which is supported by industry standards for conventional systems. There is a huge body of literature in the traditional software reliability community on OP-based testing and reliability modelling techniques, e.g., [9, 13, 53, 75]. In contrast to this, OP-based software testing for ML components is still in its infancy: to the best of our knowledge, there are only two recent works that explicitly consider the OP in testing. Li *et al.* [44] propose novel stratified sampling based on ML specific information to improve the testing efficiency. Similarly, Guerriero *et al.* [26] develop a test case sampling method that leverages "auxiliary information for misclassification" and provides unbiased reliability estimators. However, neither of them considers robustness evidence in their assessment like our RAM does.

At the whole LES level, there are reliability studies based on operational and statistical data, e.g., [37, 74] for self-driving cars, [30, 73] for AUV, and [56] for agriculture robots doing autonomous weeding. However, knowledge from low-level ML components is usually ignored. In [75], we improved [37] by providing a Bayesian mechanism to combine such knowledge, but did not discuss where to obtain the knowledge. In that sense, this article also contains follow-up work of [75], providing the prior knowledge required based on the OP and robustness evidence.

Given that the OP is essentially a distribution defined over the whole input space, a related topic is the *distribution-aware testing for DL* developed recently. For instance, in [7], distribution-guided coverage criteria are developed to guide the generation of new unseen test cases while identifying the validity of errors in DL system tasks. In [24], a generative model is utilised to guide the generation of valid test cases. However, their notion of "distribution" normally refers to *realistic* perturbations on inputs such as Gaussian noise, blur, haze, contrast variation [76], or even human imperceptible noise. Thus, it is a different notion compared to the OP that models the end-users' behaviours.

8. Discussion

8.1. Discussions on the Proposed RAM

In this section, we summarise the *model assumptions* made in our RAM, and discuss if/how they can be validated and which new assumptions and compromises in the solutions are needed to cope with real-world applications with high-dimensional data. Finally, we list the *inherent difficulties* of assessing ML reliability uncovered by our RAM.

R-Separation and its Estimation. Assumption 1 derives from Remark 3. We concur with [68] and believe that, for any real-world ML classification application where the inputs are data-points with "physical meanings", there should always exist an r-stable ground truth. Such r-stable ground truth varies between applications, and the smaller the r is, the harder the inherent difficulty of the classification problem becomes. This r is therefore a *difficulty indicator* for the given classification problem. Indeed, it is hard to estimate the r (either in the input pixel space nor the latent feature space)—the best we can do is to estimate it from the existing dataset. One way of solving the problem is to keep monitoring the r estimates as more labelled data is collected, e.g. during operation, and to redo the cell partition when the estimated r has changed significantly. Such a dynamic way of estimating r can be supported by the concept of dynamic assurance cases [2].

Approximation of the OP from Data. Assumption 2 says that the collected dataset statistically represents the OP, which may not hold for many practical reasons—e.g., when the future OP is uncertain at the training stage and data is therefore collected in a balanced way to perform well in all categories of inputs. Although we demonstrate our RAM under this assumption for simplicity, it can be easily relaxed. Essentially, we try to fit a PDF over the input space from an "operational dataset" (representing the OP). Data-points in this set can be *unlabelled* raw data generated from historical data of previous applications and simulations, which can then be scaled based on domain expert knowledge (e.g., by DL generative models that we are currently investigating). Obtaining such an operational dataset is an application-specific engineering problem, and manageable thanks to the fact that it does not require labelled data. Notably, the OP may also be approximated at *runtime* based on the data stream of operational data. Efficient KDE for data streams [54] can be used. If the OP was subject to sudden changes, change-point detectors like [70] should also be paired with the runtime estimator to robustly approximate the OP. Again, such dynamic way of estimating OP can also be supported by dynamic assurance cases [2].

Determination of the Ground Truth of a Cell. Assumptions 3 and 4 are essentially on how to determine the ground truth label for a given cell, which relates to the oracle problem of testing ML software. While this still remains challenging, we partially solve it by leveraging the r-separation property. Thanks to r, it is easy to determine a cell's ground truth when we see that it contains labelled data-points. However, for an empty cell, it is non-trivial. We assume the overall performance of the ML model is fairly good (e.g., better than a classifier doing random classifications), thus misclassifications within an empty cell are relatively rare events. We can determine the ground truth label of the cell by majority voting of predictions. Indeed, it is a strong assumption when there are some "failure regions" in the input space, within which the ML model performs really badly (even worse than random labelling). In this case, we need new mechanism to detect such "really bad failure regions" or spend more budget on, for example, asking humans to do the labelling.

Efficiency of Cell Robustness Evaluation. Although we only applied the two methods of SMC and [66] in our experiments to evaluate the local robustness, we believe that other statistical sampling methods designed for estimating the probability of rare-events could be used as well. Moreover, the cell robustness estimator in our RAM works in a "hot-swappable" manner: any new and more efficient estimator can easily be incorporated. Thus, despite being an important question, how to improve the efficiency of the robustness estimation for cells is beyond the scope of our RAM.

Conditional OP of a Cell. We assume that the distribution of inputs (the conditional OP) within each cell is uniform by Assumption 5. Although we conjecture that this is the common case due to the small size of cells (i.e., those very close/similar inputs within a small region are only subject to noise factors that can be modelled uniformly), the specific situation may vary; this requires justification in safety cases. For a real-world dataset, the conditional OP might represent certain distributions of "natural variations" [77], e.g. lighting conditions, that obey certain distributions. Ideally, the conditional OP of cells should capture the distribution of such natural variations. Recent advance on measuring the naturalness/realisticness of AEs [29] highly relates to this assumption and may relax it.

Independent λ_is and Op_is. As per Assumption 6, we assume all λ_is and Op_is are independent when "assembling" their estimates via Eqn. (11) and deriving the variance via Eqn. (12). This assumption is largely for the mathematical tractability when propagating the confidence in individual estimates at the cell-level to the *pmi*. Although this independence assumption is hard to justify in practice, it is not unusual in reliability models that do partition, e.g., in [53, 50]. We believe that RAMs are still useful under this assumption, while we envisage that Bayesian estimators leveraging joint priors and conjugacy may relax it.

Uncertainties Raised by Individual OP and Robustness Estimates. This relates to how reliable the chosen OP and robustness estimators themselves are. Our RAM is flexible and evolvable in the sense that it does not depend on any specific estimators. New and more reliable estimators can therefore easily be integrated to reduce the estimation uncertainties. Moreover, such uncertainties raised by estimators are propagated and compounded in our overall RAM results, cf. Eqn.s (12) and (15). Although we ignore them as per Assumption 7, this is arguably the case when the two estimators are fairly reliable and the number of samples k is much smaller than the sample frame size n.

Inherent Difficulties of Reliability Assessment on ML Software. Finally, based on our RAM and the discussions above, we summarise the inherent difficulties of assessing ML reliability as the following questions:

- How to accurately learn the OP in a potentially high-dimensional input space with relatively sparse data?
- How to build an accurate test oracle (to determine the ground truth label) by, e.g., leveraging the existing labels (done by humans) in the training dataset?
- What is the local distribution (i.e. the conditional OP) over a small input region (which is potentially only subject to subtle natural variations of physical conditions in the environment)?
- How to efficiently evaluate the robustness of a small region, given that AEs are normally rare events? And how to reduce the risk associated with an AE (e.g., referring to ALARP)?
- How to efficiently sample small regions from a large population (due to the high-dimensionality) of regions to test the local robustness in an unbiased and uncertainty informed way, given a limited budget?

We provide solutions in our RAM that are practical compromises (cf. Section 4.3), while the questions above are still challenging. At this stage, we doubt the existence of other RAMs for ML software with weaker assumptions that achieve the same level of rigorousness as ours, in which sense our RAM advances in this research direction.

8.2. Discussions on the Overall Assurance Case Framework and Low-Level Probabilistic Safety Arguments

With the emphasis on quantitative aspects of assuring LES (and thus complementing existing assurance frameworks, e.g., [15]), our overall assurance framework and the low-level probabilistic safety arguments together form an "vertically" end-to-end assurance case, in which a chain of safety/reliability techniques are integrated. However, the assurance case presented is still incomplete "horizontally"—some sub-cases and (side-)claims are undeveloped. Because, they are either generic claims that have been studied elsewhere (and omitted for simplicity), e.g. in [15, 4], or are still quite hard to argue in general and thus require specific expert judgement in a case-by-case manner.

The proposed safety analysis activities—HAZOP, hazard scenarios modelling, FTA, our RAM, and the determination of the system-level safety targets based on the performance of human/similar-products—are not exclusive in our assurance framework; rather we concur with [41] that credible safety cases require a heterogeneous approach. A dangerous pitfall is that those activities are not performed sufficiently because of, say, the analyser's limited engineering knowledge/experience and the lack of empirical data. This is, however, not unique to our assurance framework, but rather generic to any assurance studies.

We only present safety arguments for the classification function of the ML component, based on our new RAM for ML classifiers, leaving claims for the other three functions—localisation, detection timing, and object tracking—undeveloped[19]. The general idea and principles, however, are applicable to the other three functions, too: we may first define bespoke reliability measures for each (like *pmi* for classification), and then do probabilistic reliability modelling based on statistical testing evidence. This forms important future work.

8.3. Discussions on the Simulated AUV Case Study

So far, we have conducted a case study in simulators to validate and demonstrate our proposed methods. While defending the role of simulation in certification and regulation is beyond the scope of this work, simulation is arguably necessary for many reasons as long as the simulation satisfies some prerequisites—for example, the fidelity is justifiable, scenario-coverage is sufficiently high, and non-zero real-world testing is conducted to validate the simulation. That said, we plan to conduct a real-world case study in a physical wave tank, in which the conditions may be adjusted to have real-world disturbances, e.g., generating various types of waves in offshore scenarios and changing the lighting conditions.

[19] Certainly for real safety cases, we also need to develop claims on "non-ML" parts (e.g., capability of the development team and quality of the code) which can be addressed by conventional approaches that we omit in this work.

9. Conclusion and Future Work

This article introduces a RAM designed for ML classifiers, extending its initial version of [71] with more practical considerations for real-world applications of high-dimensional data and autonomous systems, e.g., the new estimator Eqn.s (14) and (15), alternative solutions discussed in Section 4.3, and new experiments on image datasets and an AUV mission. To the best of our knowledge, it is the first ML RAM that explicitly considers both the OP information and robustness evidence. It has also allowed us to uncover some inherent challenges when assessing ML reliability. Based on the RAM, we present probabilistic safety arguments for ML components incorporating low-level V&V evidence. To complete the "big picture", we also propose an overall assurance framework, in which a set of safety analysis activities are integrated to identify the whole LES level safety targets and break down them to component-level reliability requirements of ML functions. Finally, a case study based on simulated AUV is conducted. The case study is comprehensive in terms of exercising and demonstrating all proposed methods in our assurance framework and also identifying key challenges with recommendations for ML models of autonomous systems.

An intuitive way of perceiving our RAM, comparing with the usual accuracy testing, is that we enlarge the test set with more test cases around the "seeds" (original data-points in the test set). We determine the oracle of a new test case according to its seed's label and the r-distance. Those enlarged test results form the robustness evidence, and how much they contribute to the overall reliability is proportional to its OP. Consequently, *exposing to more tests (robustness evaluation) and being more representative of how it will be used (the OP)*, our RAM is more informative—and therefore more trustworthy. In line with the gist of our RAM, we believe that the DL reliability should follow the conceptualised equation of:

$$DL\ \ reliability = generalisability \times robustness.$$

In a nutshell, this equation says that, when assessing the reliability of ML software, we should not only consider how the DL model generalises to a new data-point (according to the future OP), but also take the local robustness around that new data-point into account.

Apart from the future work mentioned in the discussion section, we also plan to conduct more real-world case studies to examine the scalability of our methods. We presume a trained ML model for our assessment purpose. A natural follow-up question is how to actually improve the reliability when our RAM results indicate that a system is not good enough. As described in [72], we plan to investigate integrating ML debug testing (e.g. [31]) and retraining methods [6] with the RAM, to form a closed loop of debugging-improving-assessing. Last but not least, we find the idea of dynamic assurance cases [2] may have a great potential for addressing some challenges we currently face in our framework.

Appendix A. KDE Bootstrapping

Bootstrapping is a statistical approach to estimate any sampling distribution by random a sampling method. We sample with replacement from the original data points (X, Y) to obtain a new bootstrap dataset (X^b, Y^b) and train the KDE on the bootstrap dataset. Assume the bootstrapping process is repeated B times, leading to B bootstrap KDEs, denoted as $\widehat{\mathsf{Op}}^1(x), \ldots, \widehat{\mathsf{Op}}^B(x)$. Then we can estimate the variance of $\hat{f}(x)$ by the sample variance of the bootstrap KDE [22]:

$$\hat{\sigma}_B^2(x) = \frac{1}{B-1}\sum_{b=1}^{B}(\widehat{\mathsf{Op}}^b(x) - \mu_B)^2\ ,$$

where the μ_B can be approximated by

$$\hat{\mu}_B(x) = \frac{1}{B}\sum_{b=1}^{B}\widehat{\mathsf{Op}}^b(x)\ .$$

Appendix B. Details of the Yolo and VAE Models Trained in the AUV Case Study

We present more details of the Yolo and VAE models trained in the AUV case study respectively in Table B.3 and B.4, while in Figure B.20 four images reconstructed from the VAE model are shown as examples.

Table B.3: Average Precision (AP) of YOLOv3 model for object detection.

Class	Train		Test	
	AP_{50}	AP_{75}	AP_{50}	AP_{75}
Pipe	0.98343	0.73503	0.97131	0.72532
Floating Ball	0.85765	0.40094	0.90912	0.42536
Gas Canister	0.87230	0.62546	0.87406	0.60331
Gas Tank	0.98930	0.76552	0.99346	0.76824
Oil Barrel	0.84578	0.61437	0.84258	0.57856
Docking Cage	0.88771	0.32021	0.91076	0.33656
mAP	0.90603	0.57692	0.91688	0.57289

Table B.4: Reconstruction Loss and KL Divergence Loss of VAE model

VAE model	Train	Test
Recon. Loss	0.002601	0.003048
KL Div. Loss	1.732866	1.729756

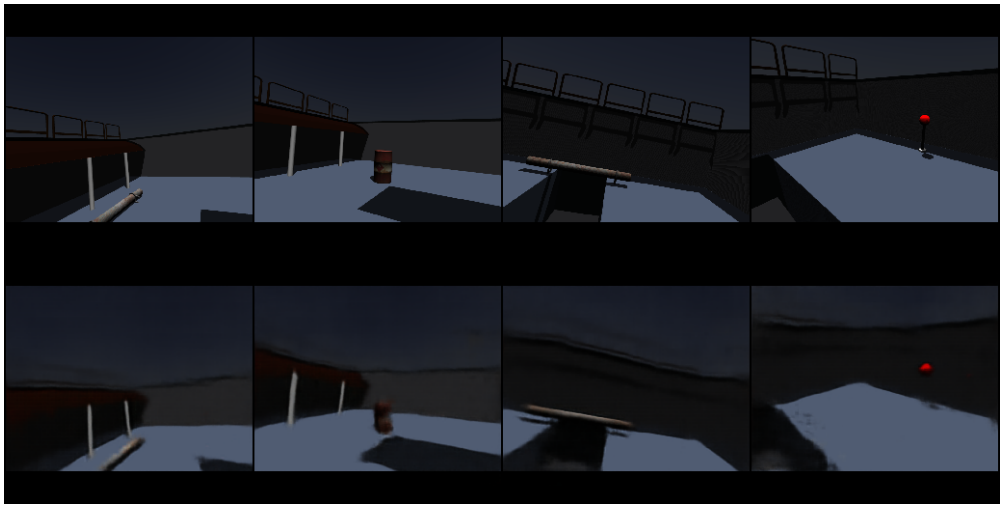

Figure B.20: Four original images (top row) and the corresponding reconstructed images (bottom row) by the VAE model.

Acknowledgments & Disclaimer

This work is supported by the UK Dstl (through the project of Safety Argument for Learning-enabled Autonomous Underwater Vehicles) and the UK EPSRC (through the Offshore Robotics for Certification of Assets [EP/R026173/1, EP/W001136/1] and End-to-End Conceptual Guarding of Neural Architectures [EP/T026995/1]). Xingyu Zhao and Alec Banks' contribution to the work is partially supported through Fellowships at the Assuring Autonomy International Programme. This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 956123. We thank Philippa Ryan for insightful comments on earlier versions of this work.

References

[1] Alves, E., Bhatt, D., Hall, B., Driscoll, K., Murugesan, A., Rushby, J., 2018. Considerations in assuring safety of increasingly autonomous systems. Technical Report NASA/CR-2018-220080. NASA.

[2] Asaadi, E., Denney, E., Menzies, J., Pai, G.J., Petroff, D., 2020a. Dynamic Assurance Cases: A Pathway to Trusted Autonomy. Computer 53, 35–46. doi:10.1109/MC.2020.3022030.

[3] Asaadi, E., Denney, E., Pai, G., 2020b. Quantifying assurance in learning-enabled systems, in: Casimiro, A., Ortmeier, F., Bitsch, F., Ferreira, P. (Eds.), Computer Safety, Reliability, and Security, Springer, Cham. pp. 270–286. doi:10.1007/978-3-030-54549-9_18.

[4] Ashmore, R., Calinescu, R., Paterson, C., 2021. Assuring the machine learning lifecycle: Desiderata, methods, and challenges. ACM Computing Surveys (CSUR) 54, 1–39.

[5] Backurs, A., Indyk, P., Wagner, T., 2019. Space and time efficient kernel density estimation in high dimensions, in: Advances in Neural Information Processing Systems, Curran Associates, Inc.. pp. 15773–15782.

[6] Bai, T., Luo, J., Zhao, J., Wen, B., Wang, Q., 2021. Recent Advances in Adversarial Training for Adversarial Robustness, in: Proc. of the 30th Int. Joint Conf. on Artificial Intelligence, pp. 4312–4321. doi:10.24963/ijcai.2021/591.

[7] Berend, D., 2021. Distribution awareness for AI system testing, in: 43rd IEEE/ACM International Conference on Software Engineering: Companion Proceedings, ICSE Companion 2021, Madrid, Spain, May 25-28, 2021, IEEE. pp. 96–98.

[8] Bergstra, J., Bengio, Y., 2012. Random search for hyper-parameter optimization. J. of Machine Learning Research 13, 281–305.

[9] Bertolino, A., Miranda, B., Pietrantuono, R., Russo, S., 2021. Adaptive Test Case Allocation, Selection and Generation Using Coverage Spectrum and Operational Profile. IEEE Transactions on Software Engineering 47, 881–898.

[10] Bevington, P.R., Robinson, D.K., Blair, J.M., Mallinckrodt, A.J., McKay, S., 1993. Data reduction and error analysis for the physical sciences. volume 7. American Institute of Physics.

[11] Bishop, P., Bloomfield, R., 2000. A methodology for safety case development. Safety and Reliability 20, 34–42.

[12] Bishop, P., Bloomfield, R., Littlewood, B., Povyakalo, A., Wright, D., 2011. Toward a formalism for conservative claims about the dependability of software-based systems. IEEE Tran. on Software Engineering 37, 708–717.

[13] Bishop, P., Povyakalo, A., 2017. Deriving a frequentist conservative confidence bound for probability of failure per demand for systems with different operational and test profiles. Reliability Engineering & System Safety 158, 246–253.

[14] Bloomfield, R., Bishop, P., 2010. Safety and assurance cases: past, present and possible future – an Adelard perspective, in: Dale, C., Anderson, T. (Eds.), Making Systems Safer, Springer London, London. pp. 51–67.

[15] Bloomfield, R., Fletcher, G., Khlaaf, H., Hinde, L., Ryan, P., 2021. Safety case templates for autonomous systems. arXiv preprint arXiv:2102.02625 .

[16] Bloomfield, R., Khlaaf, H., Conmy, P.R., Fletcher, G., 2019. Disruptive innovations and disruptive assurance: Assuring machine learning and autonomy. Computer 52, 82–89.

[17] Bloomfield, R., Netkachova, K., 2014. Building blocks for assurance cases, in: IEEE International Symposium on Software Reliability Engineering Workshops, IEEE, Naples, Italy. pp. 186–191. doi:10.1109/ISSREW.2014.72.

[18] Bloomfield, R., Rushby, J., 2020. Assurance 2.0: A manifesto. arXiv preprint arXiv:2004.10474 .

[19] Burton, S., Habli, I., Lawton, T., McDermid, J., Morgan, P., Porter, Z., 2020. Mind the gaps: Assuring the safety of autonomous systems from an engineering, ethical, and legal perspective. Artificial Intelligence 279, 103201. doi:https://doi.org/10.1016/j.artint.2019.103201.

[20] Calinescu, R., Weyns, D., Gerasimou, S., Iftikhar, M.U., Habli, I., Kelly, T., 2018. Engineering trustworthy self-adaptive software with dynamic assurance cases. IEEE Tran. on Software Engineering 44, 1039–1069.

[21] Carlini, N., Wagner, D., 2017. Towards Evaluating the Robustness of Neural Networks, in: IEEE Symp. on Security and Privacy (SP), IEEE, San Jose, CA, USA. pp. 39–57. doi:10.1109/SP.2017.49.

[22] Chen, Y.C., 2017. A tutorial on kernel density estimation and recent advances. Biostatistics & Epidemiology 1, 161–187.

[23] Crawley, F., Tyler, B., 2015. HAZOP: Guide to best practice. Elsevier.

[24] Dola, S., Dwyer, M.B., Soffa, M.L., 2021. Distribution-aware testing of neural networks using generative models, in: 43rd IEEE/ACM International Conference on Software Engineering, ICSE 2021, Madrid, Spain, 22-30 May 2021, IEEE. pp. 226–237.

[25] Guerriero, A., 2020. Reliability Evaluation of ML systems, the oracle problem, in: Int. Symp. on Software Reliability Engineering Workshops (ISSREW), IEEE, Coimbra, Portugal. pp. 127–130. doi:10.1109/ISSREW51248.2020.00050.

[26] Guerriero, A., Pietrantuono, R., Russo, S., 2021. Operation is the Hardest Teacher: Estimating DNN Accuracy Looking for Mispredictions, in: IEEE/ACM 43rd Int. Conf. on Software Engineering, Madrid, Spain. pp. 348–358. doi:10.1109/ICSE43902.2021.00042.

[27] Guo, L., Kang, J., 2015. An extended HAZOP analysis approach with dynamic fault tree. Journal of Loss Prevention in the Process Industries 38, 224–232. doi:https://doi.org/10.1016/j.jlp.2015.10.003.

[28] Hamlet, D., Taylor, R., 1990. Partition testing does not inspire confidence. IEEE Tran. on Software Engineering 16, 1402–1411.

[29] Harel-Canada, F., Wang, L., Gulzar, M.A., Gu, Q., Kim, M., 2020. Is Neuron Coverage a Meaningful Measure for Testing Deep Neural Networks?, in: Proc. of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ACM, New York, NY, USA. pp. 851–862.

[30] Hereau, A., Godary-Dejean, K., Guiochet, J., Robert, C., Claverie, T., Crestani, D., 2020. Testing an Underwater Robot Executing Transect Missions in Mayotte, in: Mohammad, A., Dong, X., Russo, M. (Eds.), Towards Autonomous Robotic Systems, Springer, Cham. pp. 116–127.

[31] Huang, W., Sun, Y., Zhao, X., Sharp, J., Ruan, W., Meng, J., Huang, X., 2021. Coverage guided testing for recurrent neural networks. IEEE Tran. on Reliability doi:0.1109/TR.2021.3080664. early access.

[32] Huang, X., Kroening, D., Ruan, W., et al, 2020. A survey of safety and trustworthiness of deep neural networks: Verification, testing, adversarial attack and defence, and interpretability. Computer Science Review 37, 100270.

[33] Huang, X., Kwiatkowska, M., Wang, S., Wu, M., 2017. Safety verification of deep neural networks, in: Computer Aided Verification, Springer International Publishing, Cham. pp. 3–29.

[34] Ishikawa, F., Matsuno, Y., 2018. Continuous argument engineering: Tackling uncertainty in machine learning based systems, in: Gallina, B., Skavhaug, A., Schoitsch, E., Bitsch, F. (Eds.), SafeComp'18, Springer, Cham. pp. 14–21.
[35] Javed, M.A., Muram, F.U., Hansson, H., Punnekkat, S., Thane, H., 2021. Towards dynamic safety assurance for Industry 4.0. Journal of Systems Architecture 114, 101914. doi:https://doi.org/10.1016/j.sysarc.2020.101914.
[36] Johnson, C. W., 2018. The increasing risks of risk assessment: On the rise of artificial intelligence and non-determinism in safety-critical systems, in: the 26th Safety-Critical Systems Symposium, Safety-Critical Systems Club, York, UK.. p. 15.
[37] Kalra, N., Paddock, S.M., 2016. Driving to safety: How many miles of driving would it take to demonstrate autonomous vehicle reliability? Transportation Research Part A: Policy and Practice 94, 182 – 193.
[38] Kelly, T.P., 1999. Arguing safety: A systematic approach to managing safety cases. PhD Thesis. University of York.
[39] Kläs, M., Adler, R., Jöckel, L., Groß, J., Reich, J., 2021. Using complementary risk acceptance criteria to structure assurance cases for safety-critical ai components, in: AISafety'21 Workshop at IJCAI'21.
[40] Knight, J., 2015. The importance of security cases: Proof is good, but not enough. IEEE Security Privacy 13, 73–75. doi:10.1109/MSP.2015.68.
[41] Koopman, P., Kane, A., Black, J., 2019. Credible autonomy safety argumentation, in: 27th Safety-Critical Systems Symp., Safety-Critical Systems Club, Bristol, UK.
[42] Lane, D., Bisset, D., Buckingham, R., Pegman, G., Prescott, T., 2016. New foresight review on robotics and autonomous systems. Technical Report No. 2016.1. Lloyd's Register Foundation. London, U.K.
[43] Lee, W.S., Grosh, D.L., Tillman, F.A., Lie, C.H., 1985. Fault Tree Analysis, Methods, and Applications - A Review. IEEE Tran. on Reliability R-34, 194–203. doi:10.1109/TR.1985.5222114.
[44] Li, Z., Ma, X., Xu, C., Cao, C., Xu, J., Lü, J., 2019. Boosting operational DNN testing efficiency through conditioning, in: Proc. of the 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ACM, New York, NY, USA. pp. 499–509. doi:10.1145/3338906.3338930.
[45] Littlewood, B., Rushby, J., 2012. Reasoning about the reliability of diverse two-channel systems in which one channel is "possibly perfect". IEEE Tran. on Software Engineering 38, 1178–1194.
[46] Littlewood, B., Strigini, L., 2000. Software reliability and dependability: A roadmap, in: Proc. of the Conference on The Future of Software Engineering, ACM, New York, NY, USA. pp. 175–188. doi:10.1145/336512.336551.
[47] Liu, P., Yang, R., Xu, Z., 2019. How safe is safe enough for self-driving vehicles? Risk Analysis 39, 315–325.
[48] Matsuno, Y., Ishikawa, F., Tokumoto, S., 2019. Tackling uncertainty in safety assurance for machine learning: Continuous argument engineering with attributed tests, in: SafeComp'19, Springer, Cham. pp. 398–404.
[49] Micouin, P., 2008. Toward a property based requirements theory: System requirements structured as a semilattice. Systems Engineering 11, 235–245.
[50] Miller, K.W., Morell, L.J., Noonan, R.E., Park, S.K., Nicol, D.M., Murrill, B.W., Voas, M., 1992. Estimating the probability of failure when testing reveals no failures. IEEE Tran. on Software Engineering 18, 33–43.
[51] Musa, J., 1993. Operational profiles in software-reliability engineering. IEEE Software 10, 14–32.
[52] Picardi, C., Hawkins, R., Paterson, C., Habli, I., 2019. A pattern for arguing the assurance of machine learning in medical diagnosis systems, in: Romanovsky, A., Troubitsyna, E., Bitsch, F. (Eds.), Computer Safety, Reliability, and Security, Springer, Cham. pp. 165–179.
[53] Pietrantuono, R., Popov, P., Russo, S., 2020. Reliability assessment of service-based software under operational profile uncertainty. Reliability Engineering & System Safety 204, 107193.
[54] Qahtan, A., Wang, S., Zhang, X., 2017. KDE-Track: An Efficient Dynamic Density Estimator for Data Streams. IEEE Tran. on Knowledge and Data Engineering 29, 642–655. doi:10.1109/TKDE.2016.2626441.
[55] Redmon, J., Farhadi, A., 2018. Yolov3: An incremental improvement. arXiv preprint arXiv:1804.02767 .
[56] Robert, C., Sotiropoulos, T., Waeselynck, H., Guiochet, J., Vernhes, S., 2020. The virtual lands of Oz: Testing an agribot in simulation. Empirical Software Engineering 25, 2025–2054. doi:10.1007/s10664-020-09800-3.
[57] Ruijters, E., Stoelinga, M., 2015. Fault tree analysis: A survey of the state-of-the-art in modeling, analysis and tools. Computer Science Review 15-16, 29–62.
[58] S. Toulmin, 1958. The Uses of Argument. Cambridge University Press.
[59] Scott, D.W., 2015. Multivariate density estimation: theory, practice, and visualization. John Wiley & Sons.
[60] Silverman, B.W., 1986. Density estimation for statistics and data analysis. volume 26. CRC press.
[61] Strigini, L., Littlewood, B., 1997. Guidelines for Statistical Testing. Technical Report. City, University of London. URL: http://openaccess.city.ac.uk/254/.
[62] Strigini, L., Povyakalo, A., 2013. Software fault-freeness and reliability predictions, in: Bitsch, F., Guiochet, J., Kaâniche, M. (Eds.), Computer Safety, Reliability, and Security, Springer Berlin Heidelberg, Berlin, Heidelberg. pp. 106–117. doi:10.1007/978-3-642-40793-2_10.
[63] Swann, C.D., Preston, M.L., 1995. Twenty-five years of HAZOPs. Journal of Loss Prevention in the Process Industries 8, 349–353. doi:https://doi.org/10.1016/0950-4230(95)00041-0.
[64] UK Office for Nuclear Regulation, 2019. The purpose, scope and content of safety cases. Nuclear Safety Technical Assessment Guide NS-TAST-GD-051. Office for Nuclear Regulation. URL: https://www.onr.org.uk/operational/tech_asst_guides/ns-tast-gd-051.pdf.
[65] Walter, G., Augustin, T., 2009. Imprecision and prior-data conflict in generalized Bayesian inference. Journal of Statistical Theory & Practice 3, 255–271.
[66] Webb, S., Rainforth, T., Teh, Y.W., Kumar, M.P., 2019. A statistical approach to assessing neural network robustness, in: 7th Int. Conf. Learning Representations (ICLR'19), OpenReview.net, New Orleans, LA, USA.
[67] Weng, L., Chen, P.Y., Nguyen, L., Squillante, M., Boopathy, A., Oseledets, I., Daniel, L., 2019. PROVEN: Verifying Robustness of Neural Networks with a Probabilistic Approach, in: Chaudhuri, K., Salakhutdinov, R. (Eds.), Proc. of the 36th Int. Conf. on Machine Learning, PMLR, Long Beach, California, USA. pp. 6727–6736.
[68] Yang, Y.Y., Rashtchian, C., Zhang, H., Salakhutdinov, R.R., Chaudhuri, K., 2020. A Closer Look at Accuracy vs. Robustness, in: Larochelle,

H., Ranzato, M., Hadsell, R., Balcan, M.F., Lin, H. (Eds.), Advances in Neural Information Processing Systems, Curran Associates, Inc.. pp. 8588–8601.

[69] Zhao, X., Banks, A., Sharp, J., Robu, V., Flynn, D., Fisher, M., Huang, X., 2020a. A Safety Framework for Critical Systems Utilising Deep Neural Networks, in: Casimiro, A., Ortmeier, F., Bitsch, F., Ferreira, P. (Eds.), Computer Safety, Reliability, and Security, Springer. pp. 244–259. doi:10.1007/978-3-030-54549-9_16.

[70] Zhao, X., Calinescu, R., Gerasimou, S., Robu, V., Flynn, D., 2020b. Interval Change-Point Detection for Runtime Probabilistic Model Checking, in: Proc. of the 35th IEEE/ACM Int. Conf. on Automated Software Engineering, ACM. pp. 163–174. doi:10.1145/3324884.3416565.

[71] Zhao, X., Huang, W., Banks, A., Cox, V., Flynn, D., Schewe, S., Huang, X., 2021a. Assessing the Reliability of Deep Learning Classifiers Through Robustness Evaluation and Operational Profiles, in: AISafety'21 Workshop at IJCAI'21.

[72] Zhao, X., Huang, W., Schewe, S., Dong, Y., Huang, X., 2021b. Detecting operational adversarial examples for reliable deep learning, in: 51th Annual IEEE-IFIP Int. Conf. on Dependable Systems and Networks (DSN'21).

[73] Zhao, X., Robu, V., Flynn, D., Dinmohammadi, F., Fisher, M., Webster, M., 2019a. Probabilistic model checking of robots deployed in extreme environments, in: Proc. of the AAAI Conference on Artificial Intelligence, Honolulu, Hawaii, USA. pp. 8076–8084.

[74] Zhao, X., Robu, V., Flynn, D., Salako, K., Strigini, L., 2019b. Assessing the safety and reliability of autonomous vehicles from road testing, in: the 30th Int. Symp. on Software Reliability Engineering, IEEE, Berlin, Germany. pp. 13–23.

[75] Zhao, X., Salako, K., Strigini, L., Robu, V., Flynn, D., 2020c. Assessing safety-critical systems from operational testing: A study on autonomous vehicles. Information and Software Technology 128, 106393.

[76] Zhao, Z., Dua, D., Singh, S., 2018. Generating natural adversarial examples, in: International Conference on Learning Representations (ICLR'18), OpenReview.net.

[77] Zhong, Z., Tian, Y., Ray, B., 2021. Understanding Local Robustness of Deep Neural Networks under Natural Variations, in: Guerra, E., Stoelinga, M. (Eds.), Fundamental Approaches to Software Engineering, Springer International Publishing, Cham. pp. 313–337.