

Reliability Assessment and Safety Arguments for Machine Learning Components in System Assurance

YI DONG and WEI HUANG*, Department of Computer Science, University of Liverpool, U.K.

VIBHAV BHARTI, School of Engineering & Physical Sciences, Heriot-Watt University, U.K.

VICTORIA COX and ALEC BANKS, Defence Science and Technology Laboratory, U.K.

SEN WANG, Department of Electrical and Electronic Engineering, Imperial College London, U.K.

XINGYU ZHAO, SVEN SCHEWE, and XIAOWEI HUANG, Department of Computer Science, University of Liverpool, U.K.

The increasing use of Machine Learning (ML) components embedded in autonomous systems—so-called Learning-Enabled Systems (LESs)—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, LESs 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 LESs 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 solutions to practical use. Probabilistic safety argument templates 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 scope our methods with case studies on simulated Autonomous Underwater Vehicles and physical Unmanned Ground Vehicles.

CCS Concepts: • **Mathematics of computing** → *Probability and statistics*; *Probability and statistics*; • **Computer systems organization** → **Reliability**; *Robotics*; **Reliability**; • **Software and its engineering** → *Fault tree analysis*; • **Computing methodologies** → **Machine learning**.

Additional Key Words and Phrases: Software reliability, safety arguments, assurance cases, safe AI, robustness verification, safety-critical systems, statistical testing, operational profile, probabilistic claims, learning-enabled systems, robotics and autonomous systems, safety regulation.

*Both authors contributed equally to this research.

Authors' addresses: Yi Dong, yi.dong@liverpool.ac.uk; Wei Huang, w.huang23@liverpool.ac.uk, Department of Computer Science, University of Liverpool, Ashton Building, Ashton Street, Liverpool, U.K., L69 3BX; Vibhav Bharti, v.bharti@hw.ac.uk, School of Engineering & Physical Sciences, Heriot-Watt University, Edinburgh, U.K., EH14 4AS; Victoria Cox, vcox@dstl.gov.uk; Alec Banks, abanks@dstl.gov.uk, Defence Science and Technology Laboratory, Salisbury, U.K., SP4 0JQX; Sen Wang, sen.wang@imperial.ac.uk, Department of Electrical and Electronic Engineering, Imperial College London, London, U.K., SW7 2BX; Xingyu Zhao, xingyu.zhao@liverpool.ac.uk; Sven Schewe, sven.schewe@liverpool.ac.uk; Xiaowei Huang, xiaowei.huang@liverpool.ac.uk, Department of Computer Science, University of Liverpool, Ashton Building, Ashton Street, Liverpool, U.K., L69 3BX.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

© 2022 Association for Computing Machinery.

Manuscript submitted to ACM

Manuscript submitted to ACM

1

ACM Reference Format:

Yi Dong, Wei Huang, Vibhav Bharti, Victoria Cox, Alec Banks, Sen Wang, Xingyu Zhao, Sven Schewe, and Xiaowei Huang. 2022. Reliability Assessment and Safety Arguments for Machine Learning Components in System Assurance. *ACM Trans. Embedd. Comput. Syst.* 1, 1 (October 2022), 47 pages. <https://doi.org/XXXXXXXX.XXXXXXX>

LIST OF ACRONYMS

ACU	Average Cell Unastuteness	MEM	Minimum Endogenous Mortality
AE	Adversarial Example	ML	Machine Learning
ALARP	As Low As Reasonably Practicable	MLE	Maximum Likelihood Estimation
AUV	Autonomous Underwater Vehicles	OP	Operational Profile
CAE	Claims Arguments Evidence	PDF	Probability Density Function
CLT	Central Limiting Theorem	PID	Proportional-Integral-Derivative
DL	Deep Learning	<i>pmi</i>	probability of misclassification per random input
DVL	Doppler Velocity Log	RAM	Reliability Assessment Models
FTA	Fault-Tree Analysis	RAS	Robotics and Autonomous Systems
GALE	Globally At Least Equivalent	ROS	Robot Operating System
GSN	Goal Structuring Notation	SE	Substantially Equivalent
HAZOP	Hazard and Operability Study	SMC	Simple Monte Carlo
IMU	Inertial Measurement Unit	UGV	Unmanned Ground Vehicle
KDE	Kernel Density Estimation	V&V	Verification and Validation
LES	Learning-Enabled Systems	VAE	Variational Auto-Encoders

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 [49]. Thus, there is an urgent need to 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) 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 [18], 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, 21, 47], despite some progress made recently [37]. 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 [16]. While inspired by [17], ours is with greater emphasis on arguing for quantitative safety requirements. This is because the unique characteristics of **ML** increase apparent non-determinism [41] that explicitly requires *probabilistic claims* to capture the uncertainties in its assurance [4, 20, 87]. 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 solution that we believe is the most practical for the moment: we exercise the method [67] which extends the **Hazard and Operability Study (HAZOP)** (a systematic hazards identification method) [78], quantitative **Fault-Tree Analysis (FTA)** (a common probabilistic root-cause analysis) [51], and propose to 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 [89] with more practical considerations for scalability. Our **RAM** explicitly takes the *Operational Profile (OP)* information and *robustness evidence* into account, because: (i) software reliability, as a *user-centred* property, depends on the end-users’ behaviours [55], and the **OP** (quantifying how the software will be operated [63]) should therefore be explicitly modelled in the assessment; (ii) a **RAM** without considering robustness evidence is not convincing given **ML** is notorious to be unrobust. 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 [33, 65], operational/statistical testing [76, 94] and **ML** robustness evaluation [82]. Our **RAM** is *model-agnostic* and designed for *pre-trained* **ML** models, yielding estimates of, e.g., expectation or confidence bounds on the metric *probability of misclassification per random input (pmi)*. Then, we present a set of safety case templates to support reliability claims¹ stated in *pmi* based on our new **RAM**—the “backbone” of the probabilistic safety arguments for **ML** components, where the key argument is over the rigour of the four main steps of the **RAM**.

Finally, we conduct comprehensive case studies based on **Autonomous Underwater Vehicles (AUV)** and **Unmanned Ground Vehicle (UGV)** in both simulated and physical environments to demonstrate and validate our methods. All source code, **ML** models, datasets and experimental results used in this work are publicly available at the our project repository <https://github.com/Solitude-SAMR> with video demos at <https://youtu.be/akY8f5sSFpY> and <https://youtu.be/E95vh5sxs7I>.

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.
- 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.

¹We 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.

- 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. Besides, real-world case studies based on a physical **UGV** and a healthcare application are conducted to support the proposed approaches.

Organisation of this Article. The scope and related work are summarised in Section 2. After presenting preliminaries in Section 3, we outline our overall *system-level* assurance framework in Section 4. We focus on the *ML-component level* where the **RAM** is described in detail in Section 5. We then present case studies in Section 6. Furthermore, in-depth discussions are provided in Section 7. Finally, we conclude and outline plans for future work in Section 8.

2 SCOPE AND RELATED WORK

The aim of this work is to bridge the gap between local robustness evaluation evidence to system-level safety claims. By the interdisciplinary nature of the target problem, it spans across different subjects including **ML**, software engineering, safety assurance and reliability engineering. We refer to Fig. 1 to highlight the scope and position of this paper (covering blocks ④, ⑤, ⑨, ⑬) as what follows.

2.1 ML Robustness Evaluation

Despite the exact definitions of robustness vary in literatures, they all share the intuition that inputs in a small region (usually a norm ball defined in some L_p -norm distance) have the same prediction label. Inside the local region, if an input is classified differently to the central point by the **ML** model, then it is detected as an **Adversarial Example (AE)**. In recent years, great efforts have been made to study the local robustness of **ML** models, while they formalise the problem in various ways and propose different evaluation metrics accordingly.

As shown in the first column of Fig. 1, the most classical type of approaches is by framing the robustness evaluation as a satisfiability problem that usually can be solved by SAT/SMT solvers [29, 38, 43, 74]. Thus, given a local perturbation distance as constraints, a binary metric (i.e., Sat/Unsat) can be evaluated (block ①). The main limitation of such methods is their scalability to the size of **ML** models and input dimensions. A more scalable type of methods is based on adversarial attacks, which normally requires an expensive gradient computation to detect **AEs** and compute a minimised perturbation distance (block ②), e.g., [2, 61, 62, 83]. Similarly, [84] converts robustness analysis into a local Lipschitz constant estimation problem, deriving a lower bound on the minimal perturbation distance required to craft an **AE**. When considering *adversarial training*, maximised prediction loss (block ③) is often the metric of interest, e.g., [57] introduces the adversarial risk which is the maximum prediction loss within norm balls to measure the local robustness. They further apply this metric as the training loss to train **ML** models that are more resistant to adversarial attacks.

At the local level, all aforementioned methods essentially answer the binary question of whether there exist any **AEs** in some perturbation distance. As argued by [82], they all suffer from two major drawbacks: i) they provide no notion of *how* robust the model is whenever an **AE** can be found; ii) there is a computational problem whenever no or only rare **AEs** exist. To address the shortfalls, [82] develops a new measure of *probabilistic* robustness (block ④) under a local distribution over a set of inputs. Arguably, such probabilistic metric is of more practical interest in twofold [82]: i) binary verification concerning extreme cases is neither necessary nor realistically achievable, i.e., one actually desires to know the *proportion* of **AEs** in the input set, not just a binary answer as to whether it is robust or not; ii) all practical applications have acceptable level of risk, so that instead of confirming that this probability is exactly zero, showing the

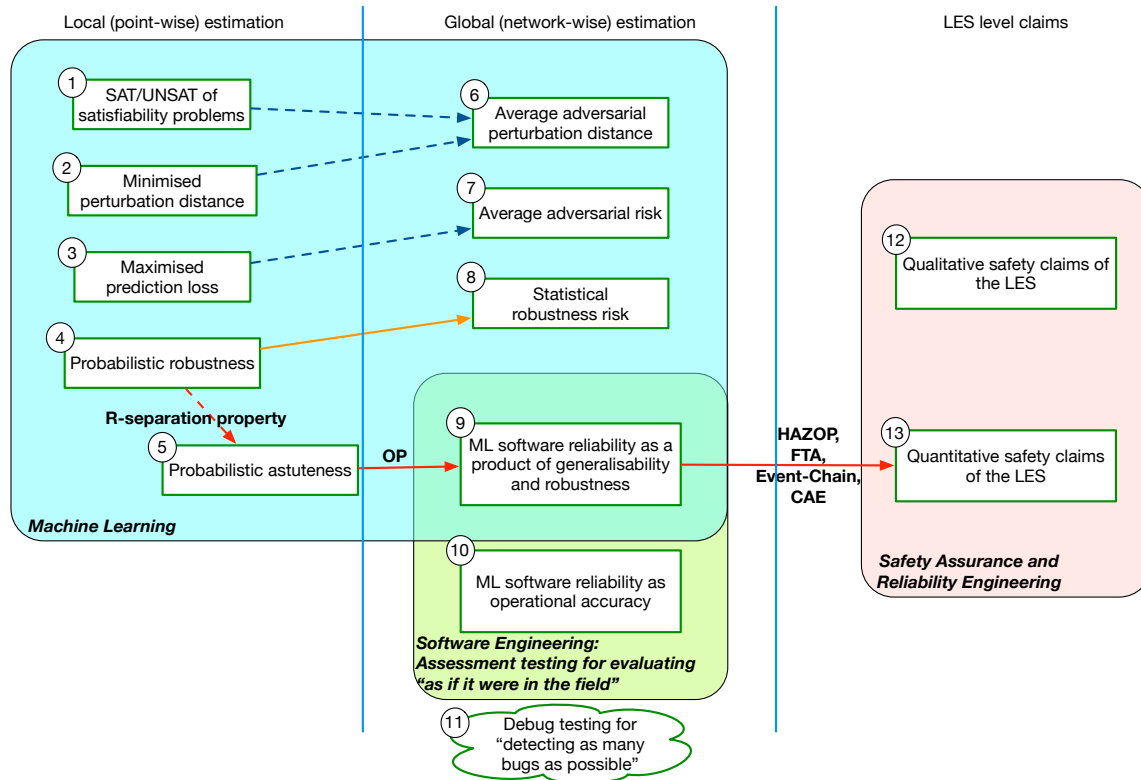


Fig. 1. Scope and position: This paper presents a chain of methods in blocks ⑤, ⑨ and ⑬ across three levels linked by techniques shown on the red edges, which has its roots in the *probabilistic* notion of local robustness (block ④). In stark contrast, other blocks at the local (i.e., point-wise robustness) level concern binary and extreme-cases that are non-probabilistic. Consequently, at the global network-wide level (i.e., the ML software), successor blocks of those non-probabilistic metrics can only consider *averaged worst-case* robustness (blocks ⑥ and ⑦). At this ML software level, our RAM (block ⑨) agrees with the statistical robustness (block ⑧) that describes the pervasiveness of AEs in the input space, but with extra considerations on the oracle and OP and thus unifies robustness (astuteness, to be exact) and generalisability. Blocks ⑨+⑩ and ⑪ are representing two distinct types of testing in software engineering, i.e. *assessment testing* vs *debug testing*, while our RAM falls into the former category making the latter category irrelevant. Unlike ours (Block ⑨), existing assessment testing methods only consider the OP and ignores the point-wise robustness evidence (Block ⑩). Finally, at the the whole system level of LES, our work supports *quantitative* safety claims.

probability of a violation below a required threshold is good enough. We concur with these key insights and extend the probabilistic robustness to probabilistic *astuteness* (block ⑤) in this work, by casting constraints (i.e., the *r*-separation property in Remark 3) on the norm ball radius as a principled way of determine the oracle (cf. later Remark 2). More explicitly, in Table 5 of Appendix C, we highlight the new characteristics of our block ⑤ by comparing with DeepFool [62] and A12 [29] locally, showing the differences between their inputs and outputs.

We usually concern the *overall* robustness of the ML model [81], i.e., its robustness across the range of possible inputs that the ML component will see in the real operation [48, 90]. This has motivated “global” or “network-wide” robustness metrics (second column in Fig. 1). Stemming from the corresponding local robustness metrics, the common metrics at this level simply take the average over a set of local robustness evaluation from regions represented by the training dataset,

e.g., average minimal adversarial distance [27], average adversarial risk [57] (blocks ⑥ and block ⑦ respectively). As discussed by [81], it is often misleading to use those metrics with binary extreme-case meanings at network-wide level—indeed, an ML model can only be perfectly worst-case robust if it is robust to all possible perturbations of all inputs, something which will very rarely be achievable in practice. Later in our experiments, we highlight the characteristics of our new block ⑨ compared to block ⑥ (that uses DeepFool [62] locally) in Table 6 of Appendix C. We may observe they are different metrics by nature, while arguably ours is of more practical interest. To overcome the problem, [81] proposes a set of statistical robustness risks (block ⑧) which assess the overall probabilistic robustness by averaging the point-wise statistical robustness of [82]. Aligning with the same idea, we build upon our local probabilistic astuteness metric to get the reliability of the ML component (block ⑨). Unlike [81], we explicitly incorporate the OP and oracle information.

2.2 OP-based Software Reliability Assessment

OP-based software testing, also known as statistical/operational testing [76], is an established practice and 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., [11, 15, 65, 94]. 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.* [52] propose novel stratified sampling based on ML specific information to improve the testing efficiency. Similarly, Guerriero *et al.* [31] develop a test case sampling method that leverages “auxiliary information for misclassification” and provides unbiased testing accuracy estimators. The motivation behind these two works is to leverage advanced sampling techniques to reduce the cost of testing in the operational dataset to yield the operational accuracy (block ⑩). 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., [42, 93] for self-driving cars, [35, 92] for AUV, and [69] for agriculture robots doing autonomous weeding. However, knowledge from low-level ML components is usually ignored. In [94], we improved [42] 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 [94], 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 Deep Learning (DL)* (block ⑪) developed recently. For instance, in [8], 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 [26], a generative model is utilised to guide the generation of valid test cases. However, all existing distribution-aware testing methods are designed for *detecting as many AEs as possible*, instead of trying to do *reliability assessment* like ours, which are two distinct types of testing [10, 24, 28].

2.3 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.* [21] 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 [18], an initial safety case framework is proposed with discussions on specific challenges for ML, which is later implemented with more details in [17]. A recent work [40] also explicitly suggests the combination of HAZOP and FTA in safety cases for identifying/mitigating hazards and deriving safety requirements (and safety contracts) when studying Industry 4.0 systems.

In [47], 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, [58] and [39], 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 [3, 22].

Although the aforementioned works have inspired this article, our assurance framework is with greater emphasis on, and thus complements them from, the quantitative aspects (block ③), 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.* [4] identifies dedicated assurance measures that are tailored for properties of aviation systems.

3 PRELIMINARIES

3.1 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 [46] and safety cases [13]. 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 [16]. 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 [79]. 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 [71]. The two most popular notations are CAE [16] and *Goal Structuring Notation (GSN)* [44]. In this article, we choose the former to present our assurance case templates.

A summary of the CAE notations is provided in Fig. 2. 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 [17, 20] 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 argumentation². These were identified by empirical analysis of real-world safety cases [19]. 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.

²The argument strategy can be either inductive or deductive [1]. For an inductive strategy, additional analysis is required to ensure that residual risks are mitigated.

An illustrative use of CAE blocks is shown in Fig. 2, while more detailed descriptions can be found in [17, 19].

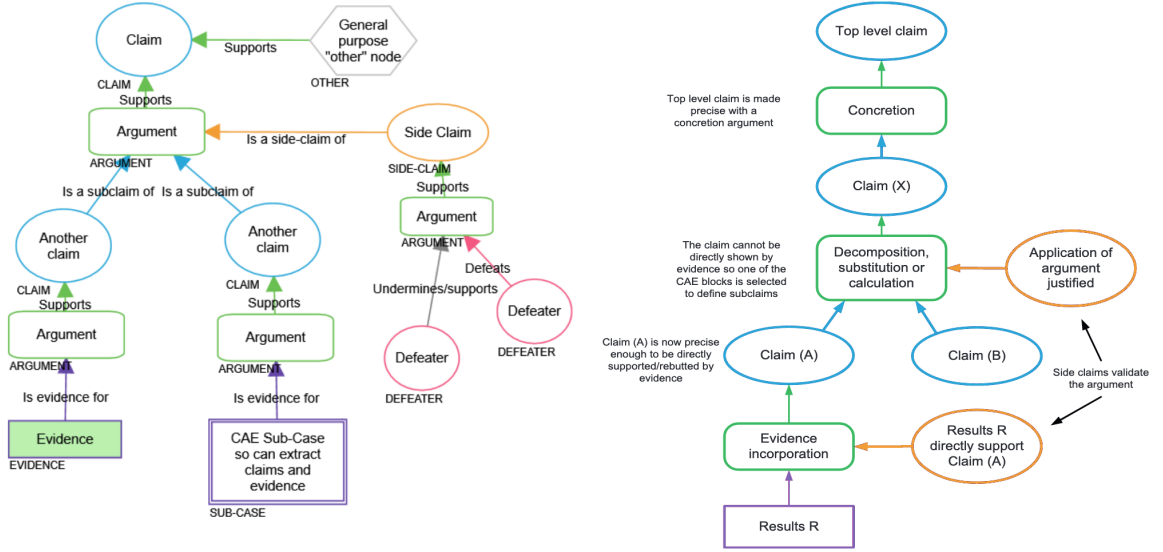


Fig. 2. Summary of the CAE notations (lhs) and an example of CAE block use (rhs), cited from [17].

3.2 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 [78]. Readers will see an illustrative example in later sections, while we refer to [25] 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 [70] for more details.

3.3 OP Based Software Reliability Assessment

The *delivered reliability*, as an *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 [55]. 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

$$\lambda := \int_{x \in X} I_{\{x \text{ causes a misclassification}\}}(x) \text{Op}(x) dx, \quad (1)$$

where x is an input in the input domain³ \mathcal{X} , and $I_S(x)$ is an indicator function—it is equal to 1 when S is true and equal to 0 otherwise. The function $\text{Op}(x)$ returns the probability that x is the next random input.

Intuitively, if we randomly selected an input from the input domain according to the distribution of **OP**, the probability of the event that this input being misclassified is measured by *pmi*. In other words, a “frequentist” interpretation of *pmi* is that it is the *limiting relative frequency* of inputs for which the classifier fails in an infinite sequence of independently selected inputs. In this regard, *pmi* is a natural extension of the conventional reliability metric *probability of failure on demand (pfd)* [91], but retrofitted for ML classifiers.

REMARK 1 (OP). *The OP [63] 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 use probability density estimators to approximate the **OP** from the collected operational dataset in our **RAM** developed in Section 5.

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 stochastic failure process, which implies a Binomial likelihood. For traditional software, 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 [14, 60], inferring ultra-high reliability [94], with certain forms of prior knowledge like perfectioness [77], with vague prior knowledge expressed in imprecise probabilities [80, 92], with uncertain **OPs** [15, 65], etc.

OP based RAMs designed for traditional software fail to consider new characteristics of **ML**, e.g., the 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 the aforementioned 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).

3.4 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 distance⁴) 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 **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. [38], while the latter is normally used in statistical approaches, e.g. [82]. The former “verification approach” is the binary version of the latter “stochastic approach”⁵.

DEFINITION 2 (ROBUSTNESS). *Similar to [82], 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):*

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

where $\text{Op}(x \mid x \in \eta)$ is the conditional **OP** of region η (precisely the “input model” used by both [82] and [83]).

³We assume continuous \mathcal{X} in this article. For discrete \mathcal{X} , the integral in Eqn. (1) reduces to sum and $\text{Op}(\cdot)$ becomes a probability mass function.

⁴Distance mentioned in this article is defined in L_∞ if without further clarification.

⁵Thus, we use the more general term robustness “evaluation” rather than robustness “verification” throughout the article.

We highlight the following 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** [85], which is also the **conditional reliability** in the region η .*

Astuteness is a special case of robustness⁶. 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.*

The r -separation property was first observed by [85]: real-world image datasets studied by the authors implies that r is normally 3 ~ 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. Fig. 3 for a conceptual illustration). Moreover, such a minimal distance is bigger than the usual norm ball size in robustness studies.

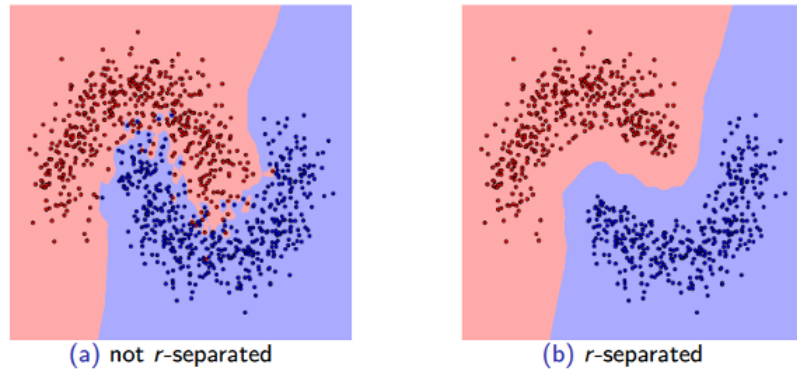


Fig. 3. Illustration of the r -separation property.

4 THE OVERALL ASSURANCE FRAMEWORK

In this section, we present an overall assurance framework for LES (e.g., AUV), in which quantitative claims stated in our metric pmi act as the bridge between point-wise robustness evidence and the whole system-level safety. As shown in Fig. 4, it starts with a dataset, e.g., the dataset used for training the ML perception component in AUV, to approximate the OP. Meanwhile, for each data-point (camera image from the AUV) in the dataset, local (point-wise) robustness can be estimated probabilistically. Then, globally at the ML component level (network-wise), a statistical model derives the pmi claim from the set of local robustness evidence considering the approximated OP. The pmi claim further supports the whole system-level safety analysis to form safety arguments in assurance cases (reusing templates for RAS extended from

⁶Thus, later in this article, we may refer to robustness as astuteness for brevity when it is clear from the context.

[17]). Since our work focuses on quantitative safety risks that can be broken down to ML-components doing perception, it *partially* implements the safety case (leaving out claims regarding other components and qualitative safety requirements). Instead of a sequential process, system level safety analysis feeds back to lower level assurance activities. That is, (i) system operational domain analysis may guide the OP approximation by pre-processing the dataset so that the dataset is statistically representing the OP; (ii) not all misclassifications are of the safety concern, thus only safety related ones should be calculated in local robustness estimation; (iii) the statistical inference model for *pmi* informs the required number of point-wise robustness estimations so that the *pmi* estimation uncertainty is sufficiently low; while (iv) the tolerable *pmi* claim is also derived from system-level safety analysis.

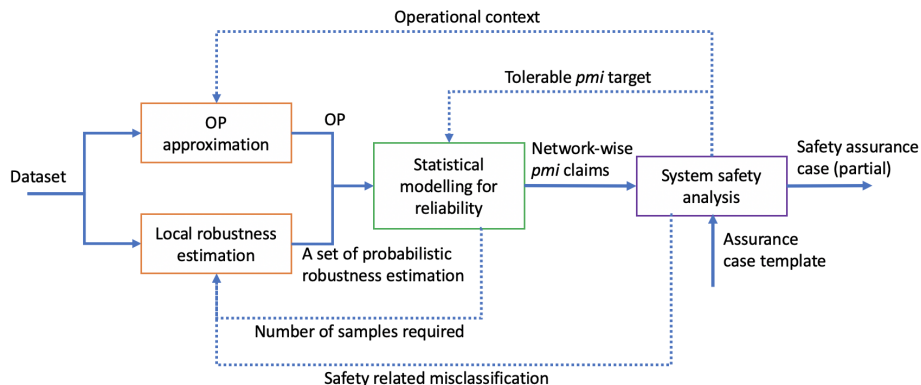


Fig. 4. Overview of the proposed assurance framework. Blocks (at different levels are in different colours) represent major assurance activities corresponding to blocks ④, ⑤, ⑨ and ⑬ in Fig. 1. Solid lines show the inputs/output of blocks from lower levels to higher levels, while dotted lines show feedbacks.

4.1 Overview of an Assurance Case for LES

The proposed assurance case template is shown in Fig. 5, which is an overview of some top-level arguments and eight supporting CAE sub-cases. It also highlights 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 of the low-level ML functionalities. It shows how various kinds of safety and reliability analysis/modelling methods are combined and structured to support our top-level claim TLC1—the LES S is acceptably safe. Additional information for the top claim should be provided (as for any safety case), describing the system S in detail and the target operational environments. The term “acceptably safe” is abstract and vague, thus we substitute it with TLC2 that all safety requirements R are satisfied. This substitution CAE-block/argument needs to be supported by a side-claim TLSC1 explaining why satisfying the set of requirements R implies S is safe enough.

To argue for TLSC1, we refer to the template proposed by [17, 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 and 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 [17], we only highlight the parts directly supporting the main focus of this work (via the procedure in Fig. 6), which are hazard identification (SubC2) and derivation of quantitative safety target (SubC3).

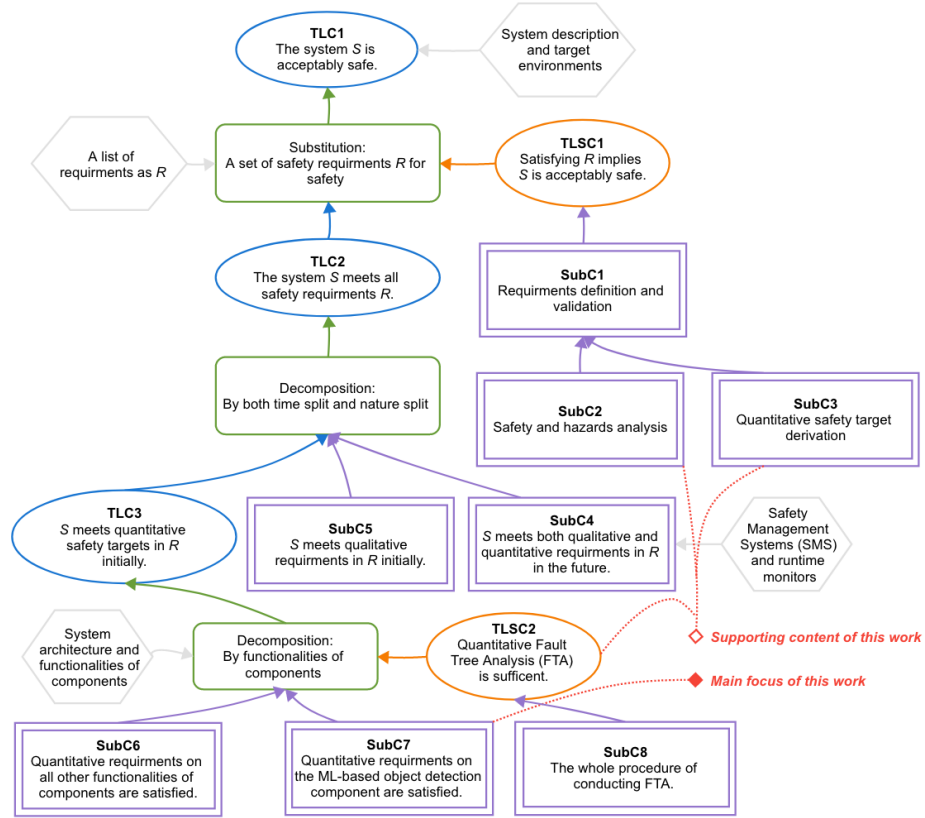


Fig. 5. Overview of the proposed safety case template for LES, highlighting the main focus and supporting content of this work.

Similar to [17], 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 Appendix E.

4.2 Deriving Quantitative Requirements for ML Components

In this work we are mainly developing low-level probabilistic safety arguments, based on the dedicated **RAM** for **ML** components developed in Section 5. 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 [53, 87]. 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. To align with this

best practice, we propose the procedure articulated in Fig. 6, whose major steps correspond to the supporting sub-cases **SubC2**, **SubC3** and **SubC8**.

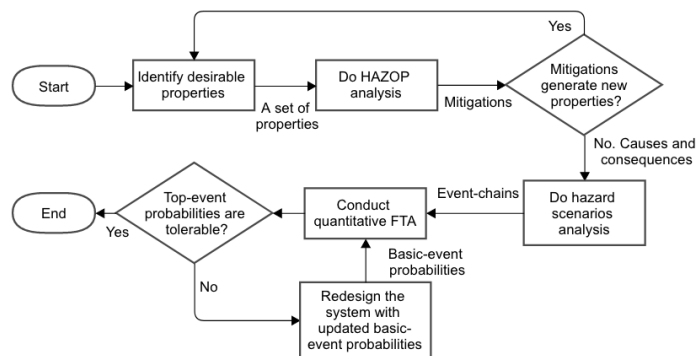


Fig. 6. The workflow of combining HAZOP and quantitative FTA to derive probabilities of basic-events of components.

In Fig. 6, 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 Fig. 6.

Then, we leverage the HAZOP results to do *hazard scenario modelling*, inspired by [32], so that we may combine HAZOP and FTA later on. Usually, as noted in [32], 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 part⁷ of the second iterative loop in Fig. 6. We refer readers to [90] for a detailed description on this *debug-retrain-assess* loop for ML software.

⁷Other non-ML components may be updated as well to jointly make the whole-system risk tolerable.

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 [18, 45]. 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, to name a few:

- **As Low As Reasonably Practicable (ALARP)**: 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**: 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.
- **Substantially Equivalent (SE)**: 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).
- **Minimum Endogenous Mortality (MEM)**: 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 [42, 56, 94]. In [64], 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.

5 MODELLING THE RELIABILITY OF ML CLASSIFIERS

5.1 A Running Example of a Synthetic Dataset

To better demonstrate our RAM, we take the Challenge of AI Dependability Assessment raised by Siemens Mobility⁸ 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. 7-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

⁸<https://ecosystem.siemens.com/topic/detail/default/33>

is labelled green, and performance related ones otherwise. For brevity, we consider both types of misclassifications here, while our RAM can cope with sub-types of misclassifications.

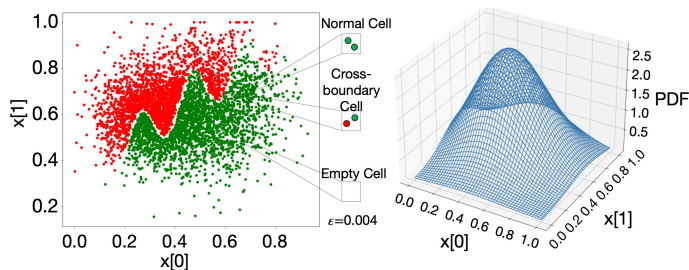


Fig. 7. The 2D-point dataset (lhs), and its approximated OP (rhs).

5.2 The Proposed RAM

Principles and Main Steps of the RAM. Inspired by [65], our RAM first partitions the input domain into m small cells⁹, subject to the r -separation property. Then, for each cell c_i (and its ground truth label y_i), we estimate:

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

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 cell¹⁰), where the weights are the pooled OP of the cells:

$$\lambda = \sum_{i=1}^m \text{Op}_i \lambda_i \quad (4)$$

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 to: (i) *how to obtain the estimates on those λ_i s and Op_i s* and (ii) *how to measure and propagate the uncertainties in the estimates*. These two questions are challenging. To name a few, for the first question: estimating λ_i requires to determine the ground truth label of cell i ; and estimating Op_i s 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_i s); (iii) evaluate the unastuteness of these cells (the λ_i s); and (iv) “assemble” all cell-wise estimates for λ in a way that estimation uncertainties are propagated and compounded.

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 the later 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 [85]) 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:

⁹We 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.

¹⁰We use “cell unastuteness” and “cell *pmi*” interchangeably later.

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 radius¹¹ 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 $\mathbb{E}[\text{Op}_i]$ and $\mathbb{V}[\text{Op}_i]$. We use the well-established **Kernel Density Estimation (KDE)** to fit a $\widehat{\text{Op}}(x)$ to approximate the OP.

ASSUMPTION 2. *The given dataset (X, Y) is collected and sampled based on the OP, and 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 relax this assumption in various ways (discussed in Section 7), we adopt it for brevity in demonstrating the running example.

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

$$\widehat{\text{Op}}(x) = \frac{1}{nh} \sum_{j=1}^n K\left(\frac{x - X_j}{h}\right), \quad (5)$$

where K is the kernel function (e.g. Gaussian or exponential kernels), and $h > 0$ is a smoothing parameter, called the bandwidth, cf. [73] for guidelines on tuning h . The approximated OP¹² is shown in Fig. 7-rhs.

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

$$\widehat{\text{Op}}_i = \widehat{\text{Op}}(x_{c_i}) v_c \quad (6)$$

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

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

where the last step of Eqn. (7) says that $\mathbb{V}[\widehat{\text{Op}}(x)]$ can be approximated using a bootstrap variance $\hat{\sigma}_B^2(x)$ [23] (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}):

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

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:

¹¹We 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.

¹²In this case, the KDE uses a Gaussian kernel and $h = 0.2$ that optimised by cross-validated grid-search [9].

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 dataset of already collected data. 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. [30]) 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 dataset size 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 will 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 [82] 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 $I_{\{M(x_j) \neq y_i\}}$. They can be approximated with sample mean $\hat{\mu}_n$ and sample variance $\hat{\sigma}_n^2/n$, respectively. Finally, we get

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

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

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 $\text{Op}(x \mid x \in c_i)$. 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 [82, 83]) and can be 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_i s and Op_i s with certainty. In practice, we can only estimate them with imperfect estimators yielding, e.g., a point estimate with variance capturing the measure of trust¹³. To assemble the estimates of λ_i s and Op_i s to get the estimates on λ , and also to propagate the confidence in those estimates, we assume:

ASSUMPTION 6. *All λ_i s and Op_i s are independent unknown variables under estimations.*

Then, the estimate of λ and its variance are:

$$\mathbb{E}[\lambda] = \sum_{i=1}^m \mathbb{E}[\lambda_i Op_i] = \sum_{i=1}^m \mathbb{E}[\lambda_i] \mathbb{E}[Op_i] \quad (11)$$

$$\mathbb{V}[\lambda] = \sum_{i=1}^m \mathbb{V}[\lambda_i Op_i] = \sum_{i=1}^m \mathbb{E}[\lambda_i]^2 \mathbb{V}[Op_i] + \mathbb{E}[Op_i]^2 \mathbb{V}[\lambda_i] + \mathbb{V}[\lambda_i] \mathbb{V}[Op_i] \quad (12)$$

Note, 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) CLT¹⁴. Then, an upper bound with $1 - \alpha$ confidence is

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

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

Complexity Analysis on RAM. The computation complexity of RAM mainly comes from the estimation of λ_i and Op_i . For each cell i , the SMC requires simulation of n_1 samples, while Op_i is estimated by KDE, trained with n_2 collected operational data. The complexity of cell-wise estimation is $\mathcal{O}(n_1 + n_2)$. To get the final estimation of DL model's λ , we assemble m cells' estimates. This results in the complexity of RAM being $\mathcal{O}(m * (n_1 + n_2))$.

5.3 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”.

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

¹³This aligns with the traditional idea of using FTA (and hence the assurance arguments around it) for future reliability assessment.

¹⁴Assuming λ_i s and Op_i s 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).

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 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 [72, 73] and the cross-validation [9]. There is also dedicated research on improving the efficiency of multivariate **KDE**, e.g., [6] 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 [82] 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., [38]. 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 [12]. 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 frame¹⁵.

¹⁵While 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 define weight w_i for each of the n norm balls according to their approximated **OP**, $w_i := \mathbb{E}[\text{Op}_i]$.
- 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 $\mathbb{E}[\lambda_i]$.
- We may invoke the unbiased estimator for weighted average [12, Chapter 4] as

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

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

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.*

Complexity Analysis on RAM Extension to High-Dimensional Dataset. For high dimensional data, RAM still adopts the KDE, fitted with n_2 operational data projected into low dimensional latent feature space. The main difference is the use of more efficient estimators for cell robustness. We refer to the Adaptive Multi-level Splitting method, an advanced Monte Carlo Simulation, that has been used for local robustness estimation in [82] and our experiments. If SMC requires the number of simulations at the order of n_1 for accurate estimation of rare events with probability $1/n_1$ (while omitting the coefficient, cf. [54] for detailed analytical results), the Adaptive Multi-level Splitting method (cf. [82] for more algorithm details) utilises the product of conditional probability with $\log n_1$ levels (the quantile ρ is normally set to 0.1). If n_3 samples are simulated by SMC for calculating each conditional probability, the computation cost of Adaptive Multi-level Splitting method is $n_3 \log n_1 \ll n_1$. Finally, we invoke the weighted sampling of k cells for cell-wise estimates assembling, the complexity of which is $O(k * (n_3 \log n_1 + n_2))$.

5.4 Evaluation on the Proposed **RAM**

In addition to the running example, we conduct experiments on two more synthetic 2D-datasets, as shown in Fig. 8. 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, as articulated in Section 5.3. 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:*

$$\text{ACU} := \frac{1}{m} \sum_{i=1}^m \lambda_i \quad (16)$$

where m is the total number of regions.

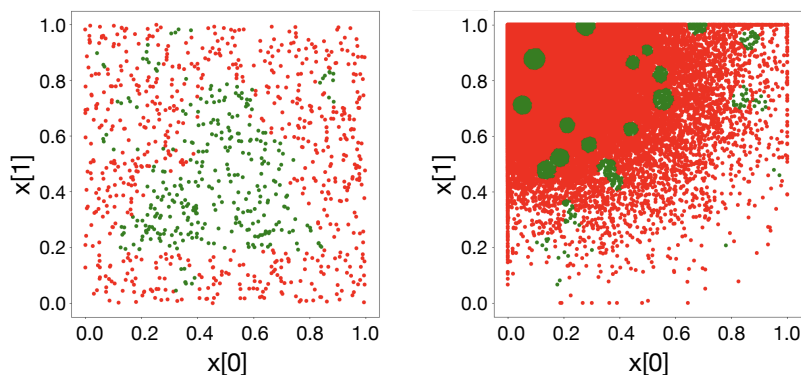


Fig. 8. Synthetic datasets DS-1 (lhs) and DS-2 (rhs) representing relatively sparse and dense training data respectively.

Table 1. Modelling details & 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
Norm. MNIST	0.0051/0.0235	0.369	0.300	k	Fig. 9(b)	Fig. 9(a)	Fig. 9(a)	Fig. 9(a)	0.43
Adv. MNIST	0.0173/0.0212	0.369	0.300	k	Fig. 9(d)	Fig. 9(c)	Fig. 9(c)	Fig. 9(c)	0.43
Norm. CIFAR10	0.0190/0.0854	0.106	0.100	k	Fig. 10(b)	Fig. 10(a)	Fig. 10(a)	Fig. 10(a)	6.74
Adv. CIFAR10	0.0013/0.1628	0.106	0.100	k	Fig. 10(d)	Fig. 10(c)	Fig. 10(c)	Fig. 10(c)	6.74

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. Fig. 7-rhs.

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 Fig. 7-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. Fig. 8-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 unrobust (by normal training) and robust (by adversarial training) ML models trained on image datasets, where the ACUs are worse and better than the test error, respectively; it confirms our aforementioned observations.

Regarding the MNIST and CIFAR10 datasets, all the experiment codes for this running example are publicly available at <https://github.com/havelhuang/ReAsDL>, and the model details are presented in Appendix B.1. In this section, we first train VAEs on them and compress the datasets into the low dimensional latent spaces of VAEs with 8 and 16 dimensions,

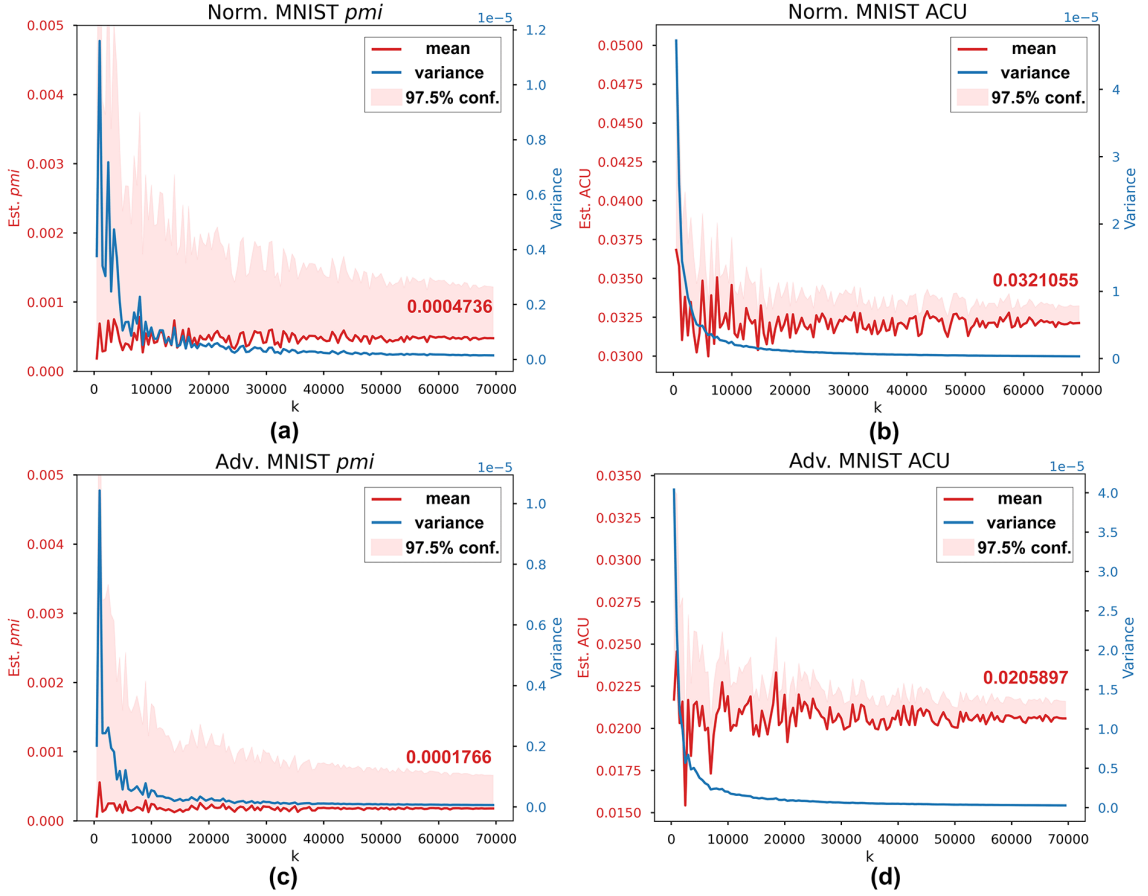


Fig. 9. The mean, variance and 97.5% confidence upper bound of pmi and ACU as functions of k sampled norm ball, estimated on MNIST dataset with normally and adversarially trained models.

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). We training two DL models with normal training strategy and PGD-based adversarial training strategy [57], respectively, and plot our RAM results for both models as functions of k in (a) and (c) of Figures 9 and 10. For comparison, we also plot the ACU results¹⁶ in (b) and (d) of Figures 9 and 10.

In Figures 9 and 10, we first observe that both the ACU results (after converging) of normally trained MNIST and CIFAR10 models 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

¹⁶As per Remark 5, ACU is a special case of pmi with equal weights. Thus, ACU results in Fig. 9, 10 are also obtained by Eqn.s (14) and (15).

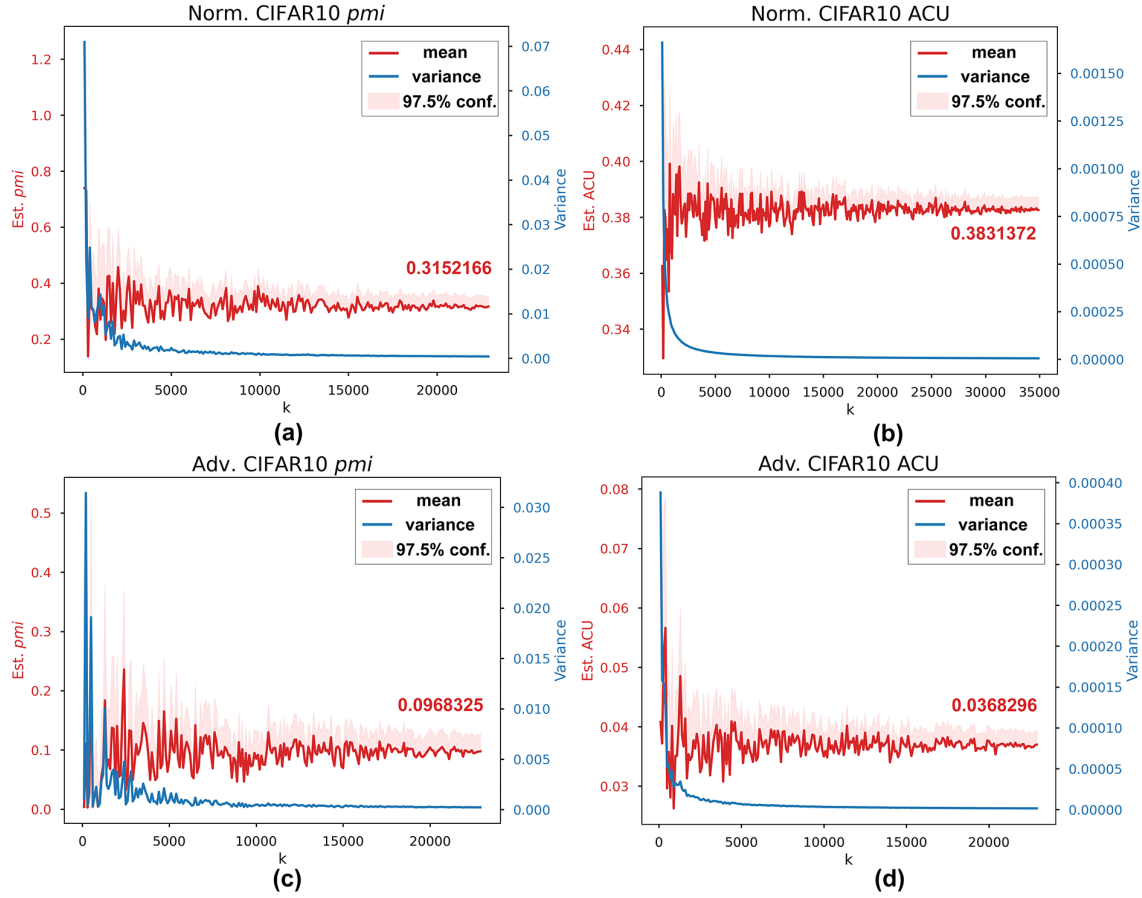


Fig. 10. The mean, variance and 97.5% confidence upper bound of pmi and ACU as functions of k sampled norm ball, estimated on CIFAR10 dataset with normally and adversarially trained models.

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). For adversarially trained models, the robustness of which is improved significantly at the cost of accuracy drop shown in Table 1. It is still effective to reduce the pmi and ACU of DL models.

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. Finally, based on the RAM, templates of probabilistic arguments for reliability claims on ML components are developed, cf. Appendix E.

6 CASE STUDIES

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 4, 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.

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 [49]. 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., [35]) 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. Fig. 6 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 Fig. 11-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.

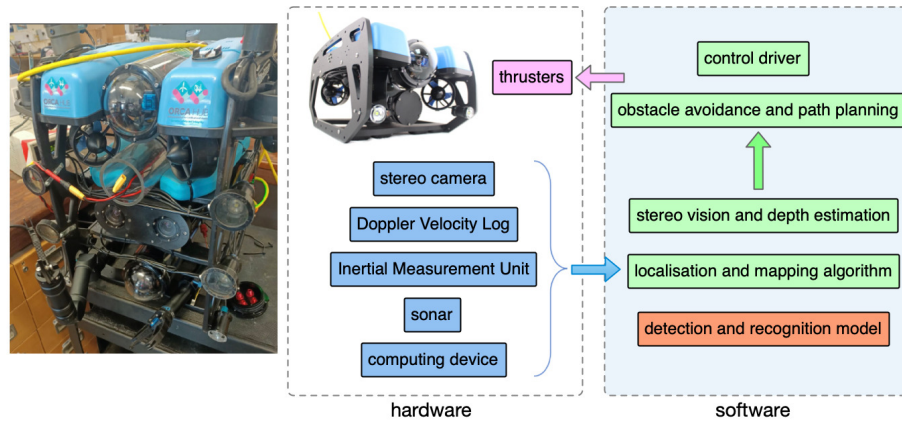


Fig. 11. Hardware–software architecture & key modules for autonomous survey & inspection missions.

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. Fig. 11):

- **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 DL architecture [68] 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. Fig. 12-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

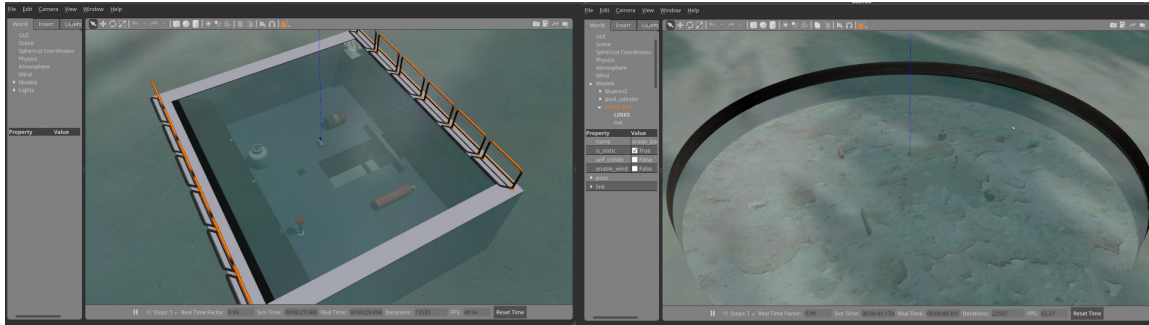


Fig. 12. A wave-tank for simulated testing and a simulated pool for collecting the training data.

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 Fig. 12-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. Fig. 13-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 Fig. 13-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.

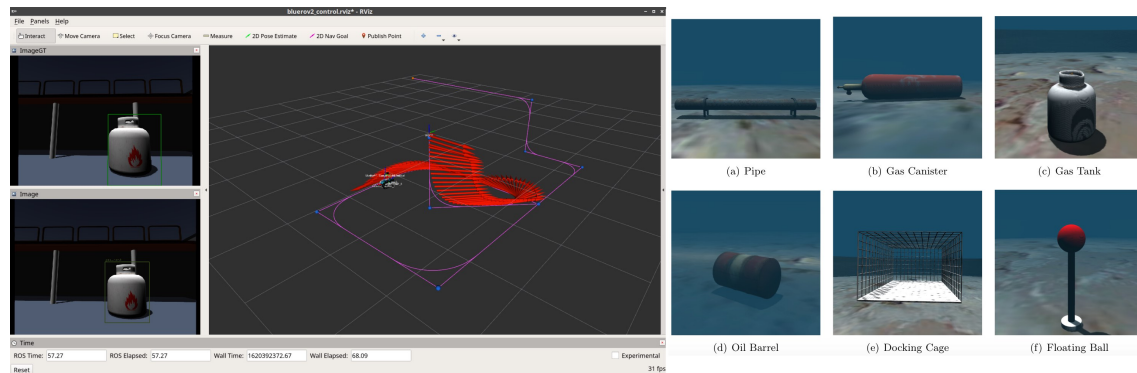


Fig. 13. Simulated AUV missions following way-points and the six simulated objects.

6.2 Assurance Activities for the AUV

Hazard Analysis via HAZOP. Given the AUV system architecture (cf. Fig. 11) and control/data flow among the nodes, we conducted a HAZOP analysis that yields a complete version of Table 2 in [67]. 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, cf. [67] and our public project repository for a complete version).

Hazard Scenarios Modelling. Inspired by [32], we 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 Fig. 14, 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 Fig. 15.

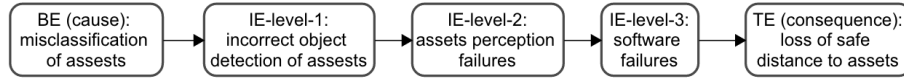


Fig. 14. An event-chain based on the hazard scenario modelling, linking causes to consequences.

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 Fig. 14 eventually becomes the path of **BE-0-1** → **IE-1-1** → **IE-2-2** → **IE-3-2** → **TE** in Fig. 15. 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 Fig. 6, 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.

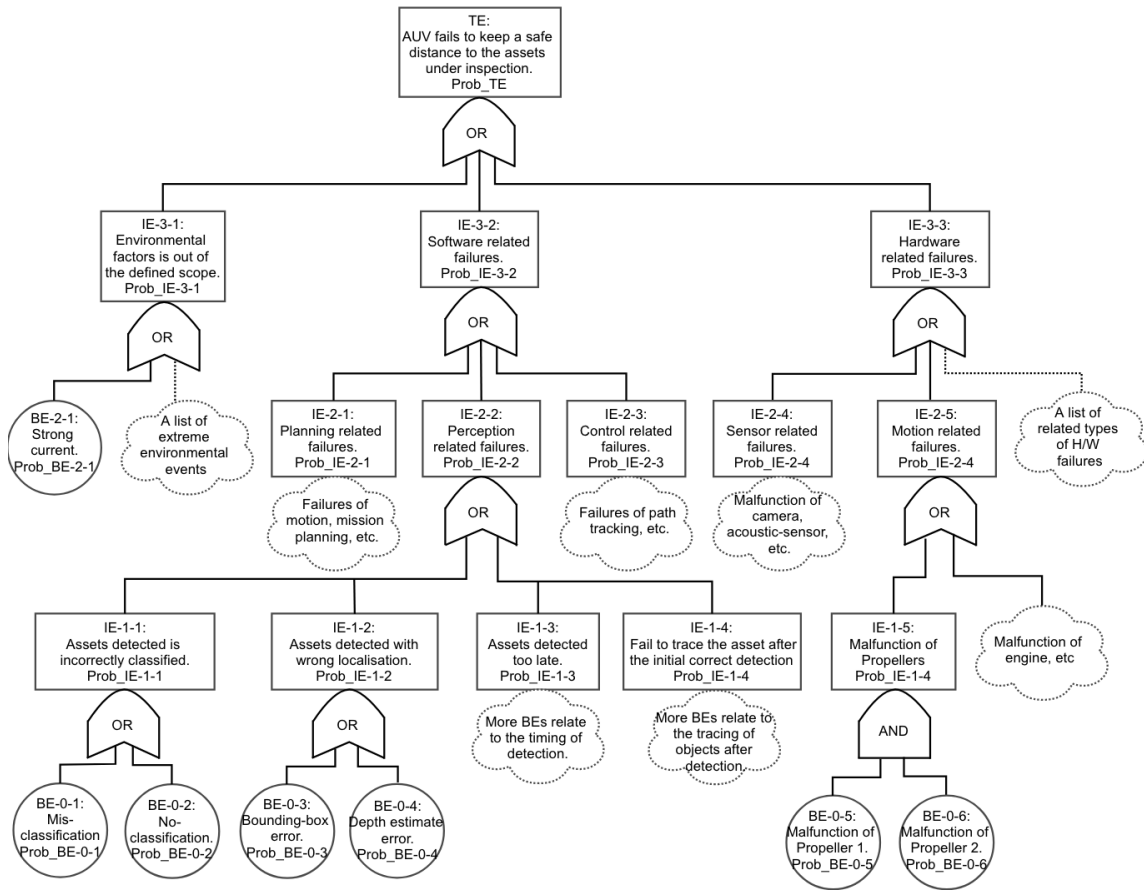


Fig. 15. A partial fault tree for the TE of loss of a safe distance to assets. NB, the “cloud” notation represents omitted sub-trees.

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 4.2 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 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 3, B.2. We adopt the practical solutions discussed in Section 5.3 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 4, four sets of examples are shown in Fig. 22, 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 distance¹⁷ and invoke the KDE and robustness estimator [82] 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 Fig. 16. Interpretation of the results is similar as before for CIFAR10, where the OP is also relatively flat. From the comparison results, it can be seen that the adversarial train can effectively improve the robustness of DL models and be captured by our RAM method.

6.4 An Extended Case Study on Real World Unmanned Ground Vehicles

In this case study, we deployed a customised real-world UGV robot, named JACKAL, as shown in Fig. 17a. A demo video is available on Youtube¹⁸.

Similar to the AUV case study, the mission of UGV case study is to detect the different traffic signs. In this experiment, we set 5 different labels, stop sign, park sign, cycle sign, cross sign and one-way sign. We trained a YOLOv3 model with normal training and adversarial training based on the same dataset. All the experiment settings and details are given about the evaluation of the RAM in Section B.2. The RAM results are described in the following parts.

As shown in Fig. 18, with the increasing amount of sampled data, the final assessment results are all converged with decreasing variance. Comparing sub-figures (a) and (b), or (c) and (d), we can find our pmi evaluation is different from the ACU in consideration of the OP. In addition, we estimate pmi and ACU on two models trained with different schemes. The train and test average precision (AP) are presented in Table 3. The adversarial training can improve the robustness of YOLO model at the cost of compromising the generalisation (indicated by mAP). The pmi metric can be conceptually perceived as a product of “generalisation \times robustness” [89]. Although previous experiments for MNIST, CIFAR10 and

¹⁷Because 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 .

¹⁸Jackal video demo: <https://youtu.be/E95vh5sxs71>

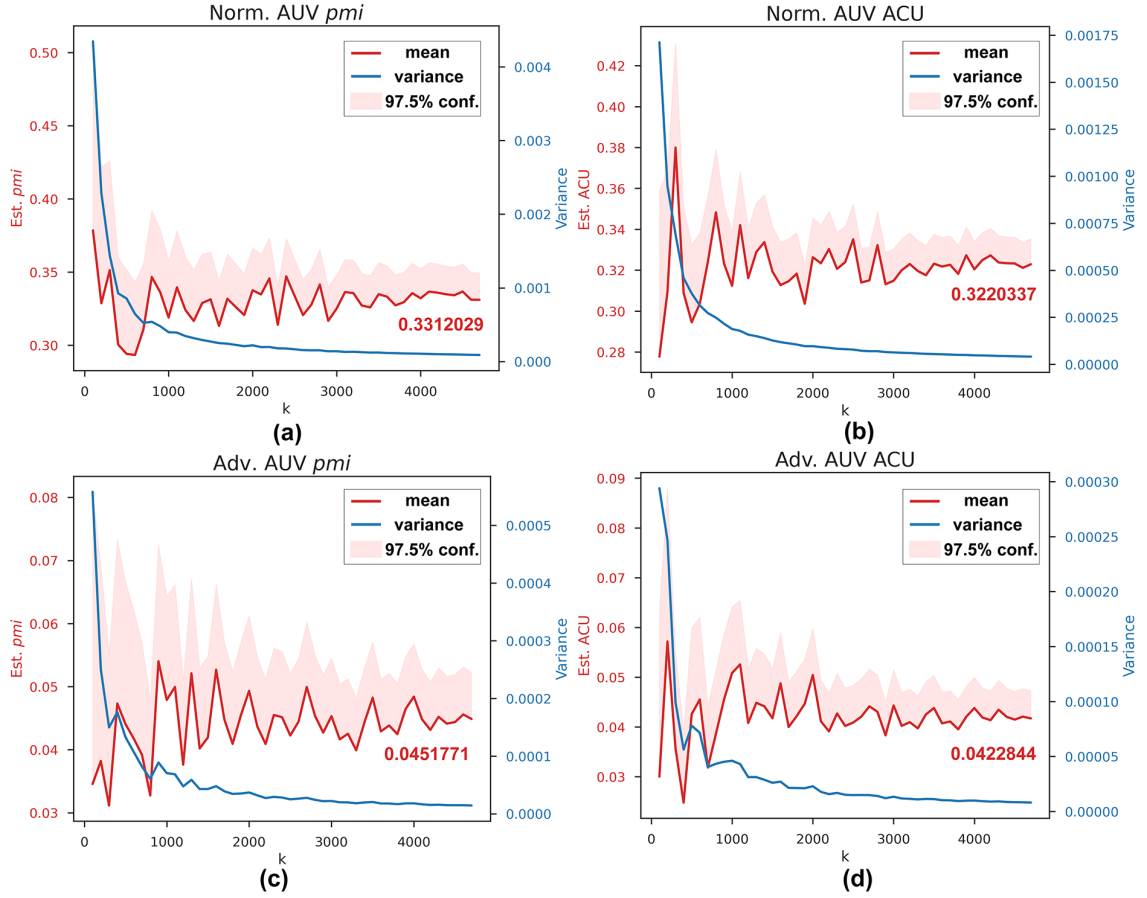


Fig. 16. The mean, variance and 97.5% confidence upper bound of AUV's pmi and ACU as functions of k sampled norm balls.

AUV shows the effectiveness of adversarial training for improving DL model's reliability by reducing the pmi , Fig. 18 displays the risk that adversarial training may potentially harm the reliability when there exists improper trade-off between generalisation and robustness. mAP for adversarially trained UGV model is significantly decreased from 0.98 to nearly 0.81, shown in Table 3. Too large drop-off of generalisation is disadvantageous to the reliability improvement. This motivates us to design new training schemes in the future towards more reliable DL models that are both robust and generalisable.

7 DISCUSSION

7.1 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.

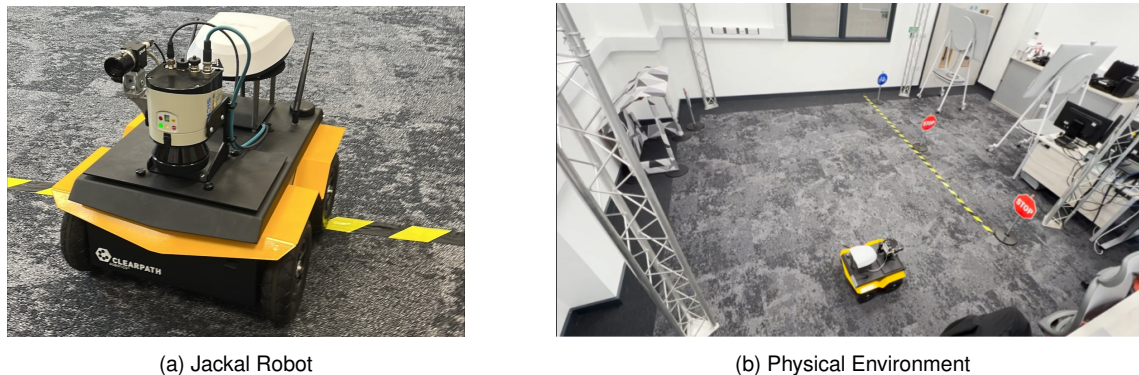


Fig. 17. The UGV and its deployed environment.

R-Separation and its Estimation. Assumption 1 derives from Remark 3. We concur with [85] 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 [3].

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 relaxed for the following reasons. Any given dataset that differs from the OP requires a *preprocessing* step to synthesis a new “operational dataset” (representing the OP), e.g., by generative models, adding weights to data-points, etc. Indeed, such preprocessing step may require domain expert knowledge and/or historical data of similar products which is not an uncommon practice for traditional software [75]. That is, we try to fit a PDF over the input space from the “operational dataset”, and data-points in this set can be raw data generated from historical data of previous applications which can then be scaled based on domain expert knowledge and by adding new data from simulation/generative-models. Obtaining such an operational dataset is an application-specific engineering problem, and manageable thanks to the fact that it does not require manual labelling. Notably, the OP may also be approximated at *runtime* based on the data stream of operational data. Efficient KDE for data streams [66] can be used. If the OP was subject to sudden changes, change-point detectors like [88] should also be paired with the runtime estimator to robustly approximate the OP. Such dynamic way of estimating OP can also be supported by dynamic assurance cases [3].

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

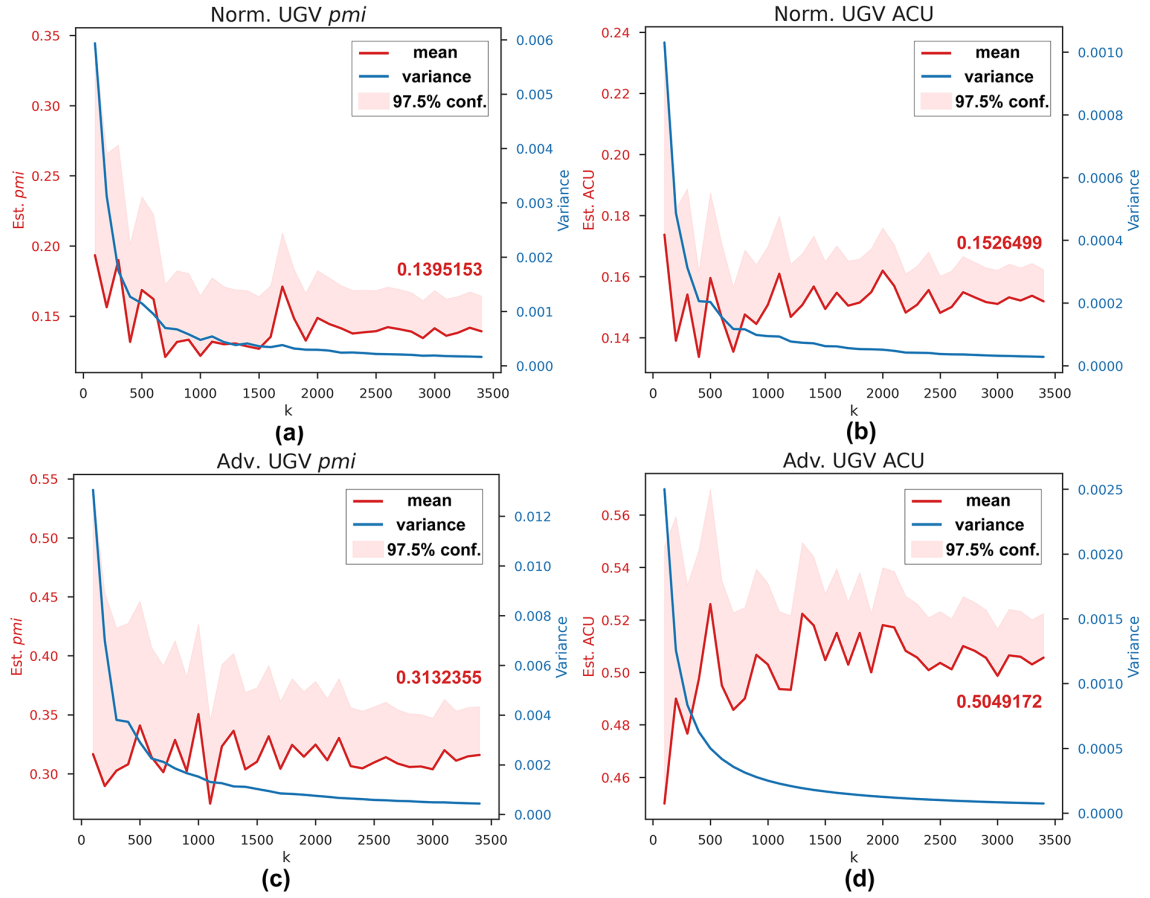


Fig. 18. The mean, variance and 97.5% confidence upper bound of UGV's pmi and ACU as functions of k sampled norm balls.

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 [82] 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” [95], 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 [34] relates to this assumption and may relax it.

Independent $\lambda_{i,s}$ and $Op_{i,s}$. As per Assumption 6, we assume all $\lambda_{i,s}$ and $Op_{i,s}$ 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 [60, 65]. 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 5.3), while the questions above are still challenging and generic. For some domains, say self-driving cars, they are relatively easier to tackle, thanks to the large amount of available data including millions of miles of historical road test data collected in recent years. 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.

7.2 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., [17]), 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 [5, 17], 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 [47] 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—undeveloped¹⁹. 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.

7.3 Discussions on the Case Studies

So far, we have conducted a AUV 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. Besides, we plan to conduct a real-world AUV 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.

To further validate and demonstrate the proposed approaches, we extended our experiments to a physical UGV case study and a CORONET case study in real-world (see Section 6.4 & Appendix D). The experimental results show that the proposed methods can acclimate to other different real-world applications.

8 CONCLUSION AND FUTURE WORK

This article introduces a RAM designed for ML classifiers, extending its initial version of [89] 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 5.3, and new experiments on image datasets, AUV/UGV missions, and a real-world healthcare application. 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.

While the main focus of this work is assessing the reliability for ML components, it is unwise to study safety without considering the LES in its application context. Because, an ML model, as software, would not cause physical mishaps on its own. What truly concerns us is the safety of the whole LES. Thus, we 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

¹⁹Certainly 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.

them to component-level reliability requirements of ML functions. Such analysis requires the understanding of the interplay between non-ML and ML components. Finally, case studies based on RAS are conducted. The case studies are comprehensive in terms of exercising and demonstrating all proposed methods in our assurance framework and also identifying key challenges with recommendations.


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, while 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:

$$DL \text{ reliability} = \text{generalisability} \times \text{robustness}.$$

Intuitively, 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 OP), but also how robustness it performs around the new data-point.

Apart from the future works 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 [90], we plan to investigate integrating ML debug testing (e.g. [36]) and retraining methods [7] with the RAM, to form a closed loop of debugging-improving-assessing. Last but not least, we find the idea of dynamic assurance cases [3] may have a great potential for addressing some challenges we currently face in our framework.

ACKNOWLEDGMENTS

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/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 and anonymous reviewers for insightful comments on earlier versions of this work. We thank TECS editors and the anonymous reviewers whose comments helped us to improve the manuscript.

This document is an overview of UK MOD (part) sponsored research and is released for informational purposes only. The contents of this document should not be interpreted as representing the views of the UK MOD, nor should it be assumed that they reflect any current or future UK MOD policy. The information contained in this document cannot supersede any statutory or contractual requirements or liabilities and is offered without prejudice or commitment.

REFERENCES

- [1] Erin Alves, Devesh Bhatt, Brendan Hall, Kevin Driscoll, Anitha Murugesan, and John Rushby. 2018. *Considerations in assuring safety of increasingly autonomous systems*. Technical Report NASA/CR-2018-220080. NASA. 172 pages.
- [2] Amir Aminifar. 2020. Universal adversarial perturbations in epileptic seizure detection. In *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 1–6.
- [3] Erfan Asaadi, Ewen Denney, Jonathan Menzies, Ganesh J. Pai, and Dimo Petroff. 2020. Dynamic Assurance Cases: A Pathway to Trusted Autonomy. *Computer* 53, 12 (2020), 35–46. <https://doi.org/10.1109/MC.2020.3022030>

- [4] Erfan Asaadi, Ewen Denney, and Ganesh Pai. 2020. Quantifying assurance in learning-enabled systems. In *Computer Safety, Reliability, and Security (LNCS)*, António Casimiro, Frank Ortmeier, Friedemann Bitsch, and Pedro Ferreira (Eds.), Vol. 12234. Springer, Cham, 270–286. https://doi.org/10.1007/978-3-030-54549-9_18
- [5] Rob Ashmore, Radu Calinescu, and Colin Paterson. 2021. Assuring the machine learning lifecycle: Desiderata, methods, and challenges. *ACM Computing Surveys (CSUR)* 54, 5 (2021), 1–39.
- [6] Arturs Backurs, Piotr Indyk, and Tal Wagner. 2019. Space and Time Efficient Kernel Density Estimation in High Dimensions. In *Advances in Neural Information Processing Systems*, Vol. 32. Curran Associates, Inc., 15773–15782.
- [7] Tao Bai, Jinqi Luo, Jun Zhao, Bihan Wen, and Qian Wang. 2021. Recent Advances in Adversarial Training for Adversarial Robustness. In *Proc. of the 30th Int. Joint Conf. on Artificial Intelligence (IJCAI'21)*. 4312–4321. <https://doi.org/10.24963/ijcai.2021/591>
- [8] David Berend. 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, 96–98.
- [9] James Bergstra and Yoshua Bengio. 2012. Random search for hyper-parameter optimization. *J. of Machine Learning Research* 13, 2 (2012), 281–305.
- [10] A. Bertolino, B. Miranda, R. Pietrantuono, and S. Russo. 2017. Adaptive Coverage and Operational Profile-Based Testing for Reliability Improvement. In *2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE)*. IEEE, Buenos Aires, Argentina, 541–551. <https://doi.org/10.1109/ICSE.2017.56>
- [11] Antonia Bertolino, Breno Miranda, Roberto Pietrantuono, and Stefano Russo. 2021. Adaptive Test Case Allocation, Selection and Generation Using Coverage Spectrum and Operational Profile. *IEEE Transactions on Software Engineering* 47, 5 (2021), 881–898.
- [12] Philip R Bevington, D Keith Robinson, J Morris Blair, A John Mallinckrodt, and Susan McKay. 1993. *Data reduction and error analysis for the physical sciences*. Vol. 7. American Institute of Physics.
- [13] Peter Bishop and Robin Bloomfield. 2000. A methodology for safety case development. *Safety and Reliability* 20, 1 (2000), 34–42.
- [14] Peter Bishop, Robin Bloomfield, Bev Littlewood, Andrey Povyakalo, and David Wright. 2011. Toward a formalism for conservative claims about the dependability of software-based systems. *IEEE Tran. on Software Engineering* 37, 5 (2011), 708–717.
- [15] Peter Bishop and Andrey Povyakalo. 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 (2017), 246–253.
- [16] Robin Bloomfield and Peter Bishop. 2010. Safety and assurance cases: past, present and possible future – an Adelard perspective. In *Making Systems Safer*, Chris Dale and Tom Anderson (Eds.). Springer London, London, 51–67.
- [17] Robin Bloomfield, Gareth Fletcher, Heidy Khlaaf, Luke Hinde, and Philippa Ryan. 2021. Safety Case Templates for Autonomous Systems. *arXiv preprint arXiv:2102.02625* (2021).
- [18] Robin Bloomfield, Heidy Khlaaf, Philippa Ryan Conmy, and Gareth Fletcher. 2019. Disruptive Innovations and Disruptive Assurance: Assuring Machine Learning and Autonomy. *Computer* 52, 9 (2019), 82–89.
- [19] R. Bloomfield and K. Netkachova. 2014. Building blocks for assurance cases. In *IEEE International Symposium on Software Reliability Engineering Workshops*. IEEE, Naples, Italy, 186–191. <https://doi.org/10.1109/ISSREW.2014.72>
- [20] Robin Bloomfield and John Rushby. 2020. Assurance 2.0: A Manifesto. *arXiv preprint arXiv:2004.10474* (2020).
- [21] Simon Burton, Ibrahim Habli, Tom Lawton, John McDermid, Phillip Morgan, and Zoe Porter. 2020. Mind the gaps: Assuring the safety of autonomous systems from an engineering, ethical, and legal perspective. *Artificial Intelligence* 279 (2020), 103201. <https://doi.org/10.1016/j.artint.2019.103201>
- [22] R. Calinescu, D. Weyns, S. Gerasimou, M. U. Iftikhar, I. Habli, and T. Kelly. 2018. Engineering trustworthy self-adaptive software with dynamic assurance cases. *IEEE Tran. on Software Engineering* 44, 11 (2018), 1039–1069.
- [23] Yen-Chi Chen. 2017. A tutorial on kernel density estimation and recent advances. *Biostatistics & Epidemiology* 1, 1 (2017), 161–187.
- [24] Domenico Cotroneo, Roberto Pietrantuono, and Stefano Russo. 2016. RELAI Testing: A Technique to Assess and Improve Software Reliability. *IEEE Transactions on Software Engineering* 42, 5 (2016), 452–475. <https://doi.org/10.1109/TSE.2015.2491931>
- [25] Frank Crawley and Brian Tyler. 2015. *HAZOP: Guide to best practice*. Elsevier.
- [26] Swaroopa Dola, Matthew B. Dwyer, and Mary Lou Soffa. 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, 226–237.
- [27] Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. 2018. Analysis of classifiers’ robustness to adversarial perturbations. *Machine learning* 107, 3 (2018), 481–508.
- [28] P. G. Frankl, R. G. Hamlet, B. Littlewood, and L. Strigini. 1998. Evaluating testing methods by delivered reliability [software]. *IEEE Tran. on Softw. Eng.* 24, 8 (1998), 586–601.
- [29] Timon Gehr, Matthew Mirman, Dana Drachler-Cohen, Petar Tsankov, Swarat Chaudhuri, and Martin Vechev. 2018. Ai2: Safety and robustness certification of neural networks with abstract interpretation. In *2018 IEEE symposium on security and privacy (SP)*. IEEE, 3–18.
- [30] Antonio Guerriero. 2020. Reliability Evaluation of ML systems, the oracle problem. In *Int. Symp. on Software Reliability Engineering Workshops (ISSREW)*. IEEE, Coimbra, Portugal, 127–130. <https://doi.org/10.1109/ISSREW51248.2020.00050>
- [31] Antonio Guerriero, Roberto Pietrantuono, and Stefano Russo. 2021. Operation is the Hardest Teacher: Estimating DNN Accuracy Looking for Mispredictions. In *IEEE/ACM 43rd Int. Conf. on Software Engineering (ICSE'21)*. Madrid, Spain, 348–358. <https://doi.org/10.1109/ICSE43902.2021.00042>
- [32] Lijie Guo and Jianxin Kang. 2015. An extended HAZOP analysis approach with dynamic fault tree. *Journal of Loss Prevention in the Process Industries* 38 (2015), 224–232. <https://doi.org/10.1016/j.jlp.2015.10.003>

- [33] D. Hamlet and R. Taylor. 1990. Partition testing does not inspire confidence. *IEEE Tran. on Software Engineering* 16, 12 (1990), 1402–1411.
- [34] Fabrice Harel-Canada, Lingxiao Wang, Muhammad Ali Gulzar, Quanquan Gu, and Miryung Kim. 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, 851–862.
- [35] Adrien Hereau, Karen Godary-Dejean, Jérémie Guiochet, Clément Robert, Thomas Claverie, and Didier Crestani. 2020. Testing an Underwater Robot Executing Transect Missions in Mayotte. In *Towards Autonomous Robotic Systems (LNCS)*, Abdelkhalick Mohammad, Xin Dong, and Matteo Russo (Eds.), Vol. 12228. Springer, Cham, 116–127.
- [36] Wei Huang, Youcheng Sun, Xingyu Zhao, James Sharp, Wenjie Ruan, Jie Meng, and Xiaowei Huang. 2021. Coverage Guided Testing for Recurrent Neural Networks. *IEEE Tran. on Reliability* (2021). <https://doi.org/10.1109/TR.2021.3080664> early access.
- [37] Xiaowei Huang, Daniel Kroening, Wenjie Ruan, and 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 (2020), 100270.
- [38] Xiaowei Huang, Marta Kwiatkowska, Sen Wang, and Min Wu. 2017. Safety verification of deep neural networks. In *Computer Aided Verification (LNCS)*, Vol. 10426. Springer International Publishing, Cham, 3–29.
- [39] Fuyuki Ishikawa and Yutaka Matsuno. 2018. Continuous argument engineering: Tackling uncertainty in machine learning based systems. In *SafeComp'18 (LNCS)*, Barbara Gallina, Amund Skavhaug, Erwin Schoitsch, and Friedemann Bitsch (Eds.), Vol. 11094. Springer, Cham, 14–21.
- [40] Muhammad Atif Javed, Faiz Ul Muram, Hans Hansson, Sasikumar Punnekkat, and Henrik Thane. 2021. Towards dynamic safety assurance for Industry 4.0. *Journal of Systems Architecture* 114 (2021), 101914. <https://doi.org/10.1016/j.sysarc.2020.101914>
- [41] 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., 15.
- [42] Nidhi Kalra and Susan M. Paddock. 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 (2016), 182 – 193.
- [43] Guy Katz, Clark Barrett, David L. Dill, Kyle Julian, and Mykel J. Kochenderfer. 2017. Reluplex: An efficient SMT solver for verifying deep neural networks. In *CAV'17 (LNCS)*, Vol. 10426. Springer, Cham, 97–117.
- [44] Timothy Patrick Kelly. 1999. *Arguing safety: A systematic approach to managing safety cases*. PhD Thesis. University of York.
- [45] Michael Kläs, Rasmus Adler, Lisa Jöckel, Janek Groß, and Jan Reich. 2021. Using Complementary Risk Acceptance Criteria to Structure Assurance Cases for Safety-Critical AI Components. In *AI Safety '21 Workshop at IJCAI'21*.
- [46] John Knight. 2015. The importance of security cases: Proof is good, but not enough. *IEEE Security Privacy* 13, 4 (2015), 73–75. <https://doi.org/10.1109/MSP.2015.68>
- [47] Philip Koopman, Aaron Kane, and Jen Black. 2019. Credible autonomy safety argumentation. In *27th Safety-Critical Systems Symp.* Safety-Critical Systems Club, Bristol, UK.
- [48] Alexey Kurakin, Ian J Goodfellow, and Samy Bengio. 2018. Adversarial examples in the physical world. In *Artificial intelligence safety and security*. Chapman and Hall/CRC, 99–112.
- [49] David Lane, David Bisset, Rob Buckingham, Geoff Pegman, and Tony Prescott. 2016. *New foresight review on robotics and autonomous systems*. Technical Report No. 2016.1. Lloyd's Register Foundation, London, U.K. 65 pages.
- [50] Rebecca J Lee, Oskar Wysocki, Cong Zhou, Rohan Shotton, Ann Tivey, Louise Lever, Joshua Woodcock, Laurence Albiges, Angelos Angelakas, Dirk Arnold, et al. 2022. Establishment of CORONET, COVID-19 Risk in Oncology Evaluation Tool, to Identify Patients With Cancer at Low Versus High Risk of Severe Complications of COVID-19 Disease On Presentation to Hospital. *JCO Clinical Cancer Informatics* 6 (2022), e2100177.
- [51] W. S. Lee, D. L. Grosh, F. A. Tillman, and C. H. Lie. 1985. Fault Tree Analysis, Methods, and Applications - A Review. *IEEE Tran. on Reliability* R-34, 3 (1985), 194–203. <https://doi.org/10.1109/TR.1985.5222114>
- [52] Zenan Li, Xiaoxing Ma, Chang Xu, Chun Cao, Jingwei Xu, and Jian Lü. 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 (ESEC/FSE 2019)*. ACM, New York, NY, USA, 499–509. <https://doi.org/10.1145/3338906.3338930>
- [53] B. Littlewood and J. Rushby. 2012. Reasoning about the reliability of diverse two-channel systems in which one channel is “possibly perfect”. *IEEE Tran. on Software Engineering* 38, 5 (2012), 1178–1194.
- [54] Bev Littlewood and Lorenzo Strigini. 1993. Validation of ultra-high dependability for software-based systems. *Commun. ACM* 36, 11 (1993), 69–80.
- [55] Bev Littlewood and Lorenzo Strigini. 2000. Software reliability and dependability: A roadmap. In *Proc. of the Conference on The Future of Software Engineering (ICSE '00)*. ACM, New York, NY, USA, 175–188. <https://doi.org/10.1145/336512.336551>
- [56] Peng Liu, Run Yang, and Zhigang Xu. 2019. How safe is safe enough for self-driving vehicles? *Risk Analysis* 39, 2 (2019), 315–325.
- [57] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2018. Towards Deep Learning Models Resistant to Adversarial Attacks. In *International Conference on Learning Representations*.
- [58] Yutaka Matsuno, Fuyuki Ishikawa, and Susumu Tokumoto. 2019. Tackling uncertainty in safety assurance for machine learning: Continuous argument engineering with attributed tests. In *SafeComp'19 (LNCS)*, Vol. 11699. Springer, Cham, 398–404.
- [59] Patrice Micouin. 2008. Toward a property based requirements theory: System requirements structured as a semilattice. *Systems Engineering* 11, 3 (2008), 235–245.
- [60] Keith W. Miller, Larry J. Morell, Robert E. Noonan, Stephen K. Park, David M. Nicol, Branson W. Murrill, and M Voas. 1992. Estimating the probability of failure when testing reveals no failures. *IEEE Tran. on Software Engineering* 18, 1 (1992), 33–43.

- [61] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. 2017. Universal adversarial perturbations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1765–1773.
- [62] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. 2016. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2574–2582.
- [63] John Musa. 1993. Operational profiles in software-reliability engineering. *IEEE Software* 10, 2 (1993), 14–32.
- [64] Chiara Picardi, Richard Hawkins, Colin Paterson, and Ibrahim Habli. 2019. A pattern for arguing the assurance of machine learning in medical diagnosis systems. In *Computer Safety, Reliability, and Security (LNCS)*, Alexander Romanovsky, Elena Troubitsyna, and Friedemann Bitsch (Eds.), Vol. 11698. Springer, Cham, 165–179.
- [65] Roberto Pietrantuono, Peter Popov, and Stefano Russo. 2020. Reliability assessment of service-based software under operational profile uncertainty. *Reliability Engineering & System Safety* 204 (2020), 107193.
- [66] Abdulhakim Qahtan, Suojin Wang, and Xiangliang Zhang. 2017. KDE-Track: An Efficient Dynamic Density Estimator for Data Streams. *IEEE Tran. on Knowledge and Data Engineering* 29, 3 (2017), 642–655. <https://doi.org/10.1109/TKDE.2016.2626441>
- [67] Yi Qi, Philippa Ryan Conmy, Wei Huang, Xingyu Zhao, and Xiaowei Huang. 2022. A Hierarchical HAZOP-Like Safety Analysis for Learning-Enabled Systems. In *AI Safety '22 Workshop at IJCAI'22*.
- [68] Joseph Redmon and Ali Farhadi. 2018. Yolo3: An incremental improvement. *arXiv preprint arXiv:1804.02767* (2018).
- [69] Clément Robert, Thierry Sotiropoulos, Hélène Waeselyncq, Jérémie Guiochet, and Simon Vernhes. 2020. The virtual lands of Oz: Testing an agribot in simulation. *Empirical Software Engineering* 25, 3 (May 2020), 2025–2054. <https://doi.org/10.1007/s10664-020-09800-3>
- [70] Enno Ruijters and Mariëlle Stoelinga. 2015. Fault tree analysis: A survey of the state-of-the-art in modeling, analysis and tools. *Computer Science Review* 15-16 (2015), 29–62.
- [71] S. Toulmin. 1958. *The Uses of Argument*. Cambridge University Press.
- [72] David W Scott. 2015. *Multivariate density estimation: theory, practice, and visualization*. John Wiley & Sons.
- [73] Bernard W Silverman. 1986. *Density estimation for statistics and data analysis*. Vol. 26. CRC press.
- [74] Gagandeep Singh, Timon Gehr, Markus Püschel, and Martin Vechev. 2019. An abstract domain for certifying neural networks. *Proceedings of the ACM on Programming Languages* 3, POPL (2019), 1–30.
- [75] Carol Smidts, Chetan Mutha, Manuel Rodríguez, and Matthew J. Gerber. 2014. Software Testing with an Operational Profile: OP Definition. *Comput. Surveys* 46, 3 (2014).
- [76] Lorenzo Strigini and Bev Littlewood. 1997. *Guidelines for Statistical Testing*. Technical Report. City, University of London. <http://openaccess.city.ac.uk/254/>
- [77] Lorenzo Strigini and Andrey Povyakalo. 2013. Software fault-freeness and reliability predictions. In *Computer Safety, Reliability, and Security (LNCS)*, Friedemann Bitsch, Jérémie Guiochet, and Mohamed Kaâniche (Eds.), Vol. 8153. Springer Berlin Heidelberg, Berlin, Heidelberg, 106–117. https://doi.org/10.1007/978-3-642-40793-2_10
- [78] C. D. Swann and M. L. Preston. 1995. Twenty-five years of HAZOPs. *Journal of Loss Prevention in the Process Industries* 8, 6 (1995), 349–353. [https://doi.org/10.1016/0950-4230\(95\)00041-0](https://doi.org/10.1016/0950-4230(95)00041-0)
- [79] 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. 39 pages. https://www.onr.org.uk/operational/tech_asst_guides/ns-tast-gd-051.pdf
- [80] Gero Walter and Thomas Augustin. 2009. Imprecision and prior-data conflict in generalized Bayesian inference. *Journal of Statistical Theory & Practice* 3, 1 (2009), 255–271.
- [81] Benjie Wang, Stefan Webb, and Tom Rainforth. 2021. Statistically robust neural network classification. In *Proc. of the 37th Conf. on Uncertainty in Artificial Intelligence*, Vol. 161. PMLR, 1735–1745.
- [82] Stefan Webb, Tom Rainforth, Yee Whye Teh, and M. Pawan Kumar. 2019. A statistical approach to assessing neural network robustness. In *7th Int. Conf. Learning Representations (ICLR'19)*. OpenReview.net, New Orleans, LA, USA.
- [83] Lily Weng, Pin-Yu Chen, Lam Nguyen, Mark Squillante, Akhilan Boopathy, Ivan Oseledets, and Luca Daniel. 2019. PROVEN: Verifying robustness of neural networks with a probabilistic approach. In *Int. Conf. on Machine Learning*. PMLR, 6727–6736.
- [84] T.-W. Weng, H. Zhang, P.-Y. Chen, J. Yi, D. Su, Y. Gao, C.-J. Hsieh, and L. Daniel. 2018. Evaluating the Robustness of Neural Networks: An Extreme Value Theory Approach. In *International Conference on Learning Representations (ICLR)*.
- [85] Yao-Yuan Yang, Cyrus Rashtchian, Hongyang Zhang, Russ R Salakhutdinov, and Kamalika Chaudhuri. 2020. A Closer Look at Accuracy vs. Robustness. In *Advances in Neural Information Processing Systems (NeurIPS'20)*, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (Eds.), Vol. 33. Curran Associates, Inc., 8588–8601.
- [86] Fisher Yu, Dequan Wang, Evan Shelhamer, and Trevor Darrell. 2018. Deep layer aggregation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2403–2412.
- [87] Xingyu Zhao, Alec Banks, James Sharp, Valentin Robu, David Flynn, Michael Fisher, and Xiaowei Huang. 2020. A Safety Framework for Critical Systems Utilising Deep Neural Networks. In *Computer Safety, Reliability, and Security (LNCS)*, António Casimiro, Frank Ortmeier, Friedemann Bitsch, and Pedro Ferreira (Eds.), Vol. 12234. Springer, 244–259. https://doi.org/10.1007/978-3-030-54549-9_16
- [88] Xingyu Zhao, Radu Calinescu, Simos Gerasimou, Valentin Robu, and David Flynn. 2020. Interval Change-Point Detection for Runtime Probabilistic Model Checking. In *Proc. of the 35th IEEE/ACM Int. Conf. on Automated Software Engineering (ASE'20)*. ACM, 163–174. <https://doi.org/10.1145/3324884.3416565>

- [89] Xingyu Zhao, Wei Huang, Alec Banks, Victoria Cox, David Flynn, Sven Schewe, and Xiaowei Huang. 2021. Assessing the Reliability of Deep Learning Classifiers Through Robustness Evaluation and Operational Profiles. In *AISafety'21 Workshop at IJCAI'21*, Vol. 2916.
- [90] Xingyu Zhao, Wei Huang, Sven Schewe, Yi Dong, and Xiaowei Huang. 2021. Detecting Operational Adversarial Examples for Reliable Deep Learning. In *51th Annual IEEE-IFIP Int. Conf. on Dependable Systems and Networks (DSN'21)*, Vol. Fast Abstract.
- [91] Xingyu Zhao, Bev Littlewood, Andrey Povyakalo, Lorenzo Strigini, and David Wright. 2017. Modeling the probability of failure on demand (pfd) of a 1-out-of-2 system in which one channel is “quasi-perfect”. *Reliability Engineering & System Safety* 158 (2017), 230–245.
- [92] Xingyu Zhao, Valentin Robu, David Flynn, Fateme Dinmohammadi, Michael Fisher, and Matt Webster. 2019. Probabilistic model checking of robots deployed in extreme environments. In *Proc. of the AAAI Conference on Artificial Intelligence*, Vol. 33. Honolulu, Hawaii, USA, 8076–8084.
- [93] Xingyu Zhao, Valentin Robu, David Flynn, Kizito Salako, and Lorenzo Strigini. 2019. Assessing the safety and reliability of autonomous vehicles from road testing. In *the 30th Int. Symp. on Software Reliability Engineering*. IEEE, Berlin, Germany, 13–23.
- [94] Xingyu Zhao, Kizito Salako, Lorenzo Strigini, Valentin Robu, and David Flynn. 2020. Assessing safety-critical systems from operational testing: A study on autonomous vehicles. *Information and Software Technology* 128 (2020), 106393.
- [95] Ziyuan Zhong, Yuchi Tian, and Baishakhi Ray. 2021. Understanding Local Robustness of Deep Neural Networks under Natural Variations. In *Fundamental Approaches to Software Engineering (LNCS)*, Esther Guerra and Mariëlle Stoeltinga (Eds.), Vol. 12649. Springer International Publishing, Cham, 313–337.

A KDE BOOTSTRAPPING

Bootstrapping is a statistical approach to estimate any sampling distribution by a random 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 $\widehat{\text{Op}}^1(x), \dots, \widehat{\text{Op}}^B(x)$. Then we can estimate the variance of $\hat{f}(x)$ by the sample variance of the bootstrap KDE [23]:

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

where the μ_B can be approximated by

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

B EXPERIMENT SETUP

In this section, the details of different experiments are summarised. All the cases are performed using Python on a computer equipped with an AMD core EPYC 7452 and NVIDIA’s GPU A100.

B.1 Details of the MNIST and CIFAR10 Models

Fig. 19 and 21 present the architecture of DNNs trained on MNIST and CIFAR10 datasets. The Deep Layer Aggregation (DLA) model [86] is especially used for the CIFAR10 dataset to achieve very high train and test accuracy.

MNIST dataset contains 70000 handwritten digit images, among which 60000 images are used for training while 10000 are used for testing the model’s generalisation. We use Adam optimiser with learning rate 10^{-3} , cross-entropy loss as training loss function and train 20 epochs to get the result.

CIFAR10 dataset contains 60000 32x32 colour images in 10 classes, with 6000 images per class. There are 50000 train images and 10000 test images. We use Stochastic gradient descent (SGD) optimiser with a learning rate of 0.01, momentum 0.9, and weight decay 5×10^{-4} to train the model with 50 epochs.

We adopt adversarial training to obtain a more robust model for reliability comparison. We use PGD-based adversarial training [57] with the same training parameters as normal training to generate attacked data (AEs when successfully attacked) in replacement of all original training data to improve robustness. To be specific, the PGD attack calculates the

Layer (type)	Output Shape	Param #
Linear-1	[-1, 1, 256]	200,960
ReLU-2	[-1, 1, 256]	0
Linear-3	[-1, 1, 10]	2,570

Total params: 203,530
Trainable params: 203,530
Non-trainable params: 0

Fig. 19. The architecture of DNN trained on MNIST dataset

Layer (type:depth-idx)	Output Shape	Param #
Sequential: 1-1	[-1, 32, 4, 4]	--
Sequential: 2-1	[-1, 8, 16, 16]	--
Conv2d: 3-1	[-1, 8, 16, 16]	224
BatchNorm2d: 3-2	[-1, 8, 16, 16]	16
ReLU: 3-3	[-1, 8, 16, 16]	--
Sequential: 2-2	[-1, 16, 8, 8]	--
Conv2d: 3-4	[-1, 16, 8, 8]	1,168
BatchNorm2d: 3-5	[-1, 16, 8, 8]	32
ReLU: 3-6	[-1, 16, 8, 8]	--
Conv2d: 2-3	[-1, 32, 4, 4]	4,640
Linear: 1-2	[-1, 16]	8,208
Linear: 1-3	[-1, 16]	8,208
Linear: 1-4	[-1, 512]	8,704
Sequential: 1-5	[-1, 3, 32, 32]	--
Sequential: 2-4	[-1, 16, 8, 8]	--
ConvTranspose2d: 3-7	[-1, 16, 8, 8]	8,208
BatchNorm2d: 3-8	[-1, 16, 8, 8]	32
ReLU: 3-9	[-1, 16, 8, 8]	--
Sequential: 2-5	[-1, 8, 16, 16]	--
ConvTranspose2d: 3-10	[-1, 8, 16, 16]	2,056
BatchNorm2d: 3-11	[-1, 8, 16, 16]	16
ReLU: 3-12	[-1, 8, 16, 16]	--
ConvTranspose2d: 2-6	[-1, 3, 32, 32]	387
Sigmoid: 2-7	[-1, 3, 32, 32]	--

Total params: 41,899
Trainable params: 41,899
Non-trainable params: 0
Total mult-adds (M): 1.70

Fig. 20. The architecture of convolutional variational autoencoder (VAE)

Layer (type:depth-idx)	Output Shape	Param #
Sequential: 1-1	[-1, 16, 32, 32]	--
Conv2d: 2-1	[-1, 16, 32, 32]	432
BatchNorm2d: 2-2	[-1, 16, 32, 32]	32
ReLU: 2-3	[-1, 16, 32, 32]	--
Sequential: 1-2	[-1, 16, 32, 32]	--
Conv2d: 2-4	[-1, 16, 32, 32]	2,304
BatchNorm2d: 2-5	[-1, 16, 32, 32]	32
ReLU: 2-6	[-1, 16, 32, 32]	--
Sequential: 1-3	[-1, 32, 32, 32]	--
Conv2d: 2-7	[-1, 32, 32, 32]	4,608
BatchNorm2d: 2-8	[-1, 32, 32, 32]	64
ReLU: 2-9	[-1, 32, 32, 32]	--
Tree: 1-4	[-1, 64, 32, 32]	--
BasicBlock: 2-10	[-1, 64, 32, 32]	--
Conv2d: 3-1	[-1, 64, 32, 32]	18,432
BatchNorm2d: 3-2	[-1, 64, 32, 32]	128
Conv2d: 3-3	[-1, 64, 32, 32]	36,864
BatchNorm2d: 3-4	[-1, 64, 32, 32]	128
Sequential: 3-5	[-1, 64, 32, 32]	2,176
BasicBlock: 2-11	[-1, 64, 32, 32]	--
Conv2d: 3-6	[-1, 64, 32, 32]	36,864
BatchNorm2d: 3-7	[-1, 64, 32, 32]	128
Conv2d: 3-8	[-1, 64, 32, 32]	36,864
BatchNorm2d: 3-9	[-1, 64, 32, 32]	128
Sequential: 3-10	[-1, 64, 32, 32]	--
Root: 2-12	[-1, 64, 32, 32]	--
Conv2d: 3-11	[-1, 64, 32, 32]	8,192
BatchNorm2d: 3-12	[-1, 64, 32, 32]	128
Tree: 1-5	[-1, 128, 16, 16]	--
Tree: 2-13	[-1, 128, 16, 16]	--
BasicBlock: 3-13	[-1, 128, 16, 16]	230,144
BasicBlock: 3-14	[-1, 128, 16, 16]	295,424
Root: 3-15	[-1, 128, 16, 16]	33,024
Tree: 2-14	[-1, 128, 16, 16]	--
BasicBlock: 3-16	[-1, 128, 16, 16]	295,424
BasicBlock: 3-17	[-1, 128, 16, 16]	295,424
Root: 3-18	[-1, 128, 16, 16]	33,024
Root: 2-15	[-1, 128, 16, 16]	--
Conv2d: 3-19	[-1, 128, 16, 16]	32,768
BatchNorm2d: 3-20	[-1, 128, 16, 16]	256
Tree: 1-6	[-1, 256, 8, 8]	--
Tree: 2-16	[-1, 256, 8, 8]	--
BasicBlock: 3-21	[-1, 256, 8, 8]	919,040
BasicBlock: 3-22	[-1, 256, 8, 8]	1,180,672
Root: 3-23	[-1, 256, 8, 8]	131,584
Tree: 2-17	[-1, 256, 8, 8]	--
BasicBlock: 3-24	[-1, 256, 8, 8]	1,180,672
BasicBlock: 3-25	[-1, 256, 8, 8]	1,180,672
Root: 3-26	[-1, 256, 8, 8]	131,584
Root: 2-18	[-1, 256, 8, 8]	--
Conv2d: 3-27	[-1, 256, 8, 8]	131,072
BatchNorm2d: 3-28	[-1, 256, 8, 8]	512
Tree: 1-7	[-1, 512, 4, 4]	--
BasicBlock: 2-19	[-1, 512, 4, 4]	--
Conv2d: 3-29	[-1, 512, 4, 4]	1,179,648
BatchNorm2d: 3-30	[-1, 512, 4, 4]	1,024
Conv2d: 3-31	[-1, 512, 4, 4]	2,359,296
BatchNorm2d: 3-32	[-1, 512, 4, 4]	1,024
Sequential: 3-33	[-1, 512, 4, 4]	132,096
BasicBlock: 2-20	[-1, 512, 4, 4]	--
Conv2d: 3-34	[-1, 512, 4, 4]	2,359,296
BatchNorm2d: 3-35	[-1, 512, 4, 4]	1,024
Conv2d: 3-36	[-1, 512, 4, 4]	2,359,296
BatchNorm2d: 3-37	[-1, 512, 4, 4]	1,024
Sequential: 3-38	[-1, 512, 4, 4]	--
Root: 2-21	[-1, 512, 4, 4]	--
Conv2d: 3-39	[-1, 512, 4, 4]	524,288
BatchNorm2d: 3-40	[-1, 512, 4, 4]	1,024
Linear: 1-8	[-1, 10]	5,130

Total params: 15,142,970
Trainable params: 15,142,970
Non-trainable params: 0
Total mult-adds (M): 945.64

Fig. 21. The architecture of Deep Layer Aggregation model trained on CIFAR10 dataset

gradient of training loss²⁰ with respect to the input. It is deployed with 10 steps and 2/255 step size to generate pixel-level perturbation for training data. The perturbed training data will be utilised over the whole training process.

We use the VAE model to reduce the dimension of data, the architecture of which is shown in Fig. 20. To train the VAE model, we use an Adam optimiser with a learning rate 10^{-3} and train 100 epochs.

²⁰We usually use cross entropy loss for classification problems. The training loss for object detection models like YoloV3 includes the objectness score of a bounding box.

B.2 Details of the YOLOv3 and VAE Models Trained in the Case Studies

We present more details of the YOLOv3 and VAE models trained in the AUV and UGV case studies, respectively in Table 3 and 4, while in Fig. 22 eight images reconstructed based on the VAE models are shown as examples. For more detailed ML models, datasets and experimental results, please refer to our public repository <https://github.com/Solitude-SAMR>.

Class	Normal Training		Adversarial Training		Class	Normal Training		Adversarial Training	
	Train	Test	Train	Test		Train	Test	Train	Test
Pipe	0.98343	0.97131	0.92225	0.91380	Stop Sign	0.97958	0.99471	0.76873	0.79065
Floating Ball	0.85765	0.90912	0.83785	0.88291	Park Sign	0.98182	0.98158	0.84459	0.82143
Gas Canister	0.87230	0.87406	0.83245	0.84319	Cycle Sign	0.99911	0.98648	0.92750	0.88669
Gas Tank	0.98930	0.99346	0.93623	0.93769	Cross Sign	0.99067	0.98878	0.98850	0.98496
Oil Barrel	0.84578	0.84258	0.82811	0.82973	One Way Sign	0.98921	0.99023	0.53972	0.52041
Docking Cage	0.88771	0.91076	0.72786	0.71478	mAP	0.98808	0.98836	0.81381	0.80083
mAP	0.90603	0.91688	0.84746	0.85368					

(a) AUV

(b) UGV

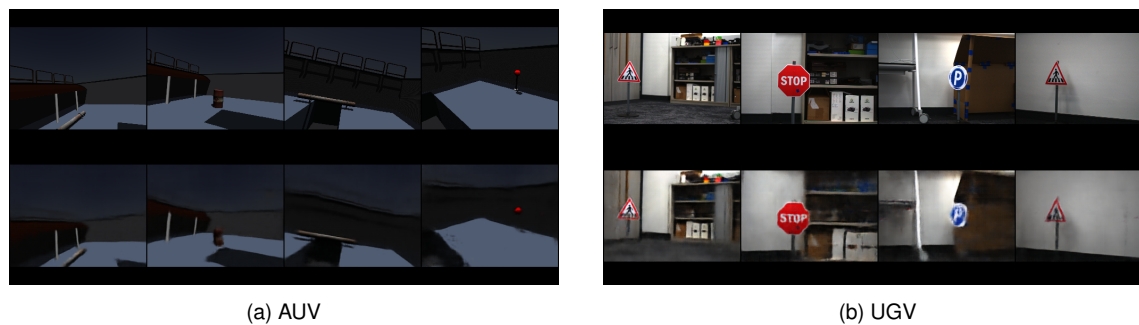
Table 3. Average Precision (AP) of YOLOv3 for object detection.

VAE model	Train	Test	VAE model	Train	Test
Recon. Loss	0.002601	0.003048	Recon. Loss	0.011333	0.016665
KL Div. Loss	1.732866	1.729756	KL Div. Loss	4.781964	4.810094

(a) AUV

(b) UGV

Table 4. Reconstruction Loss & KL Divergence Loss of VAE model



(a) AUV

(b) UGV

Fig. 22. Original images (top) and corresponding reconstructed images (bottom) by the VAE model.

C COMPARISON BETWEEN RAM AND STATE-OF-THE-ART ROBUSTNESS METRICS

As shown in Table 5, as before, we study both normally and adversarially trained DL models on the two datasets. In our RAM, the norm ball radius ϵ is determined by the r -separation distance and the local probabilistic robustness evaluation is returned by the estimator [82]. For a fair comparison, we reuse the same norm ball for applying AI2 which indeed returns the answer “unverified” to the binary question. In contrast, DeepFool aims at finding a minimal perturbation

Table 5. Local robustness evaluation around a selected test seed

	Model	RAM		DeepFool		ERAN (AI2)	
		Radius ϵ	$\hat{\lambda}_i$	Radius ϵ	Minimal Perturbation	Radius ϵ	Verified Robustness
MNIST	Norm.	0.3	5.85×10^{-19}	/	0.3151	0.3	0
	Adv.	0.3	1.93×10^{-22}	/	0.7912	0.3	0
CIFAR-10	Norm.	0.1	0.4177	/	0.0189	0.1	0
	Adv.	0.1	1.92×10^{-22}	/	0.2145	0.1	0

Table 6. Network-wise reliability and robustness evaluation

	Model	RAM			DeepFool		
		Radius ϵ	k	pmi	Radius ϵ	k	Avg. Minimal Perturbation
MNIST	Norm.	0.3	70000	4.74 e-4	/	70000	0.1979
	Adv.	0.3	70000	1.77 e-4	/	70000	0.5141
CIFAR-10	Norm.	0.1	25000	0.3152	/	60000	0.0318
	Adv.	0.1	25000	0.0968	/	60000	0.3368

to detect AEs around the same seed. Although DeepFool formulates it as an optimisation problem, its solution has no guarantees to be optimal—the returned solutions are larger than the r -separation distance when the AEs are quite rare (the 1st, 2nd and 4th rows). While, at the global level of networks, Table 6 “compares” the reliability metric pmi of our RAM with the *averaged* minimal perturbation by applying DeepFool. We omit the experiments based on AI2 at such a network-wise level due to the high computational cost of formally verifying all k norm balls. Both the assessments by RAM and DeepFool are reflective of the robustness improvement from a normally trained model to an adversarially trained model. The averaged minimal perturbation derived by DeepFool (or other similar tools concerning worst-case metrics) has limited practical values [81, 82]. At the same time, our reliability pmi concerns the overall robustness and generalisability, that predicts the probability of failure for the next input drawn from the OP.

D ADDITIONAL CASE STUDY ON HEALTHCARE MODELS

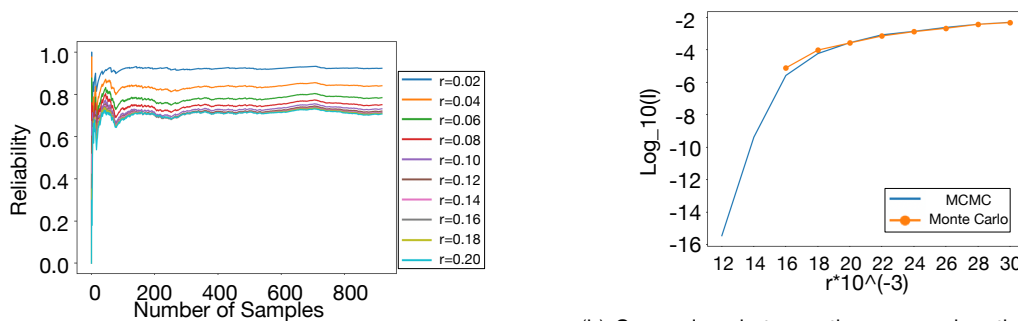
*CORONET*²¹ is a decision-support tool validated in healthcare systems worldwide can aid admission decisions and predict COVID-19 severity in patients with cancer [50]. The dependability of healthcare AI/ML systems needs to be assured to increase their trustworthiness. In this section, a case of applying our approach on *CORONET* is studied to validate our reliability assessment methods.

The *CORONET* model is a decision tree model with 11 dimensions inputs, and three different output labels. Here, we first assess the reliability of a given *CORONET* model among different r -separation radius. The experiment result is shown in Fig. 23a.

As can be seen from Fig. 23a, the reliability of the *CORONET* model decreases as the disturbance radius increases, which is in accordance with our intuition. The higher disturbance would cause more uncertainty.

To further evaluate the effectiveness of our reliability assessment model, we compare our method with naive Monte-Carlo sampling methods. The results are shown in Fig. 23b.

²¹<https://coronet.manchester.ac.uk/>



(a) PMI of CORONET dataset with different norm ball radius.

(b) Comparison between the proposed method and Monte-Carlo sampling.

Fig. 23. Experiment results of CORONET.

In this experiment, we collect 10^5 samples based on naive Monte-Carlo sampling method to assess the reliability of the *CORONET* prediction model. From the experiment results, we can see that the proposed method has an equivalent estimate to the unbiased estimation results of Monte-Carlo sampling. Furthermore, the proposed method can make it much easier to find rare events than the naive Monte Carlo method when the disturbance radius is tiny.

E PROBABILISTIC SAFETY ARGUMENTS FOR ML COMPONENTS

At this lower level of ML components, cf. the **SubC7** in Fig. 5, 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 in details, while focusing more on the latter.

E.1 Arguments for Top Claims on Object Detection at the ML Component-Level

In Fig. 24, 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 [59], 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 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

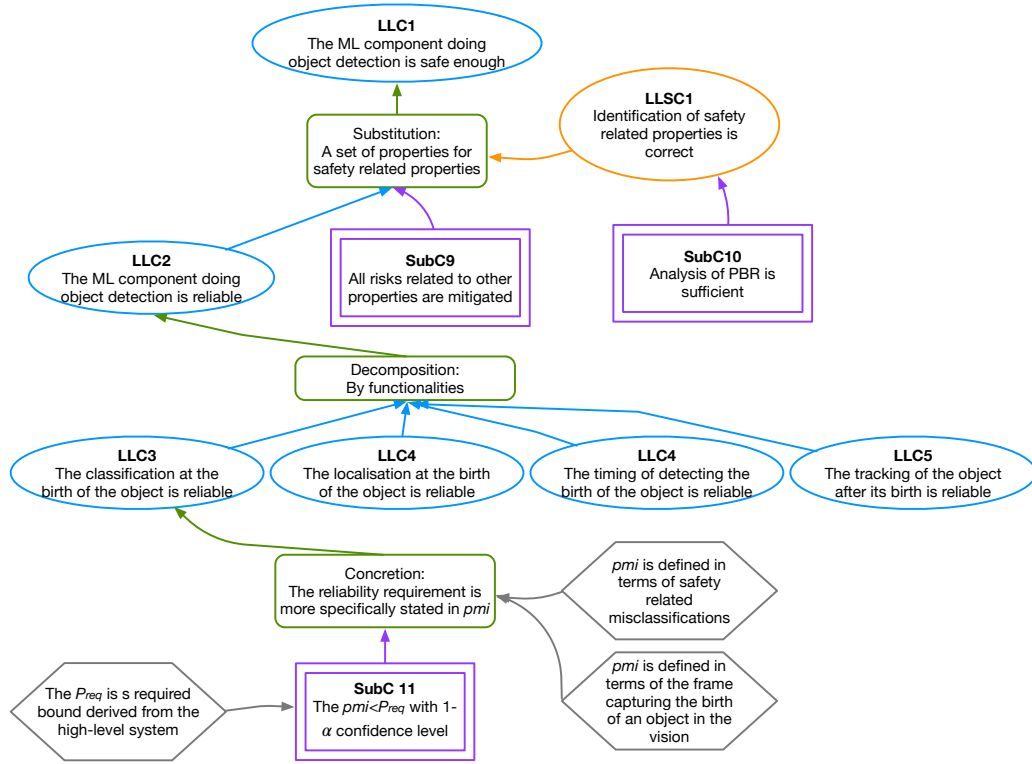


Fig. 24. ML component-level arguments breaking down the claim “The object detection component is safe enough” **LLC1** to reliability claims of the classification function stated in specific reliability measures **SubC11**.

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 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.

E.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 5—the “backbone” of the probabilistic arguments at this lower level. Essentially, we argue over the four main steps of our **RAM** as shown in Fig. 25. 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 sparse 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 examine exhaustively 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).

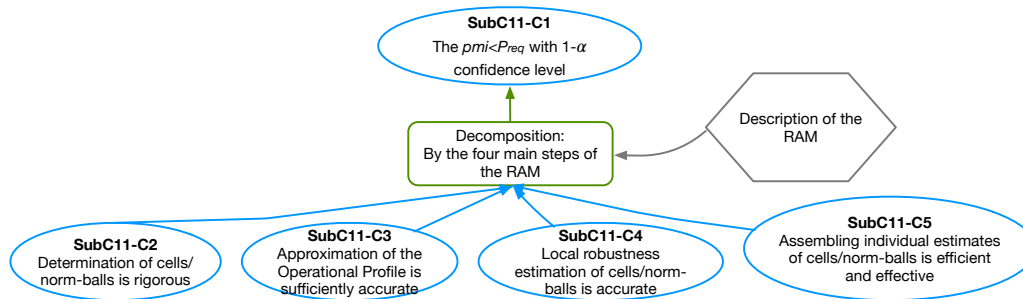


Fig. 25. Arguments over the four main steps in the proposed RAM.

Figures 26 to 29 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 5, 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 is omitted for brevity, because they are generic claims and thus can be referred to other works, e.g. [5], for **SubC11-C3.2** and **SubC11-C3.3** when we treat the OP estimator as a common data-driven learning model.

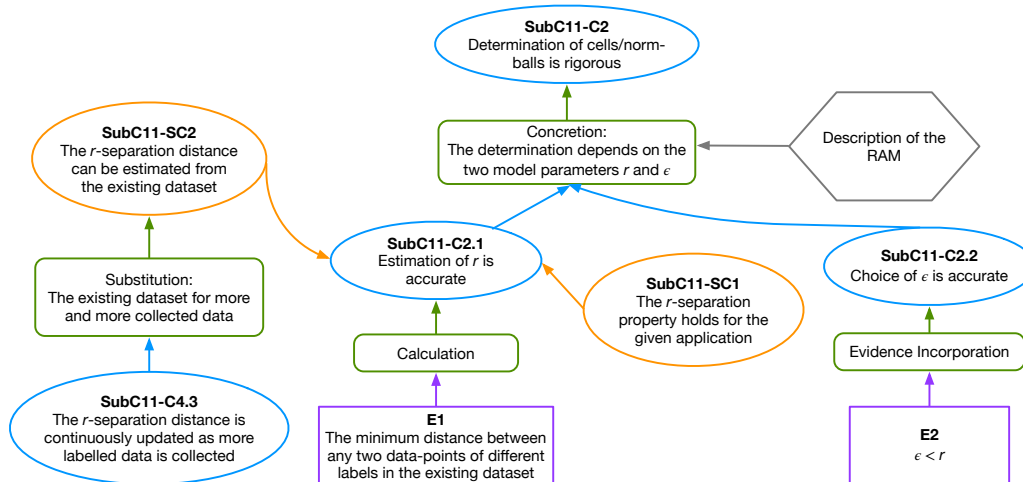


Fig. 26. Arguments based on the step 1 of the RAM.

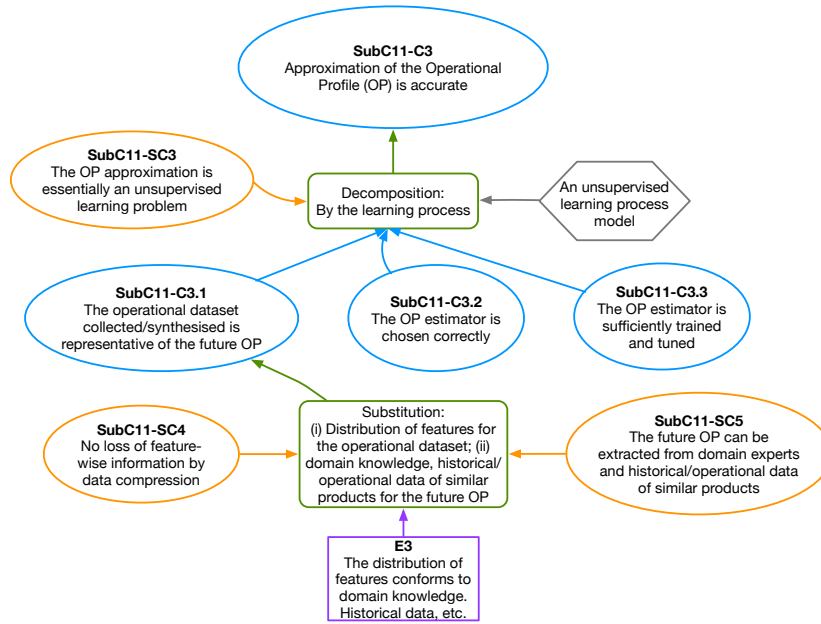


Fig. 27. Arguments based on the step 2 of the RAM.

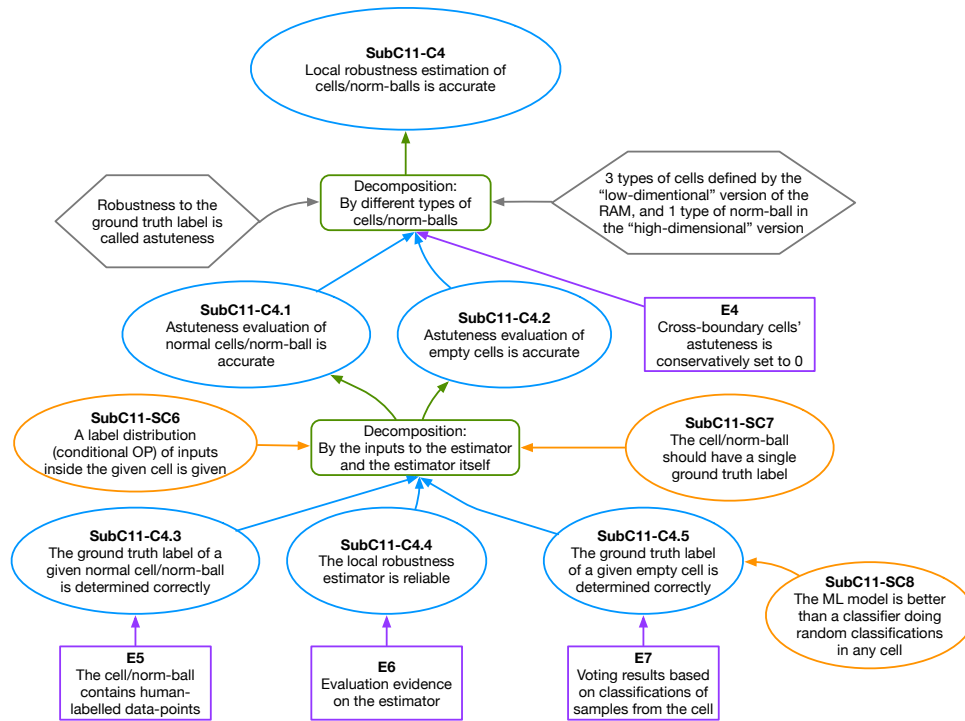


Fig. 28. Arguments based on the step 3 of the RAM.

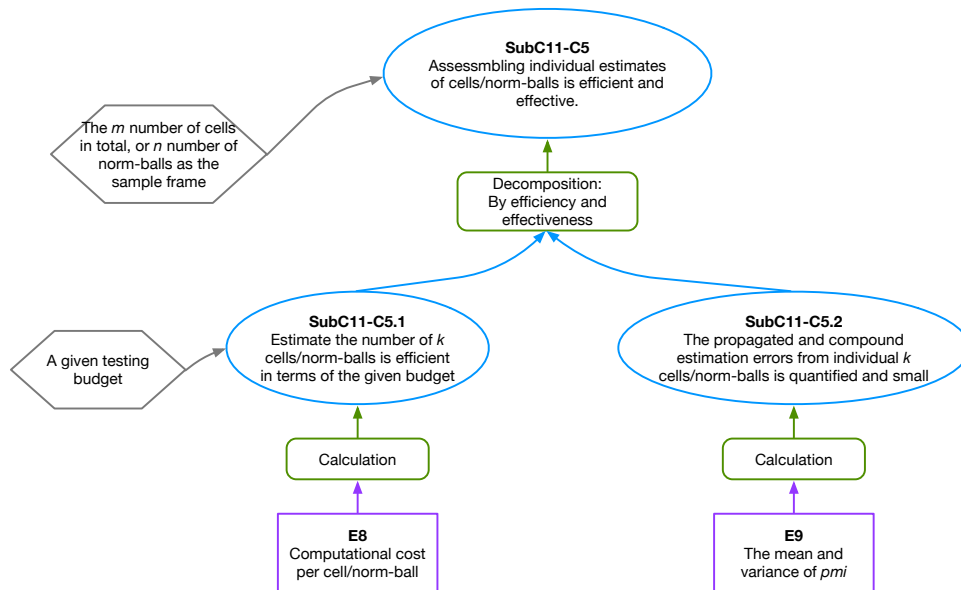


Fig. 29. Arguments based on the step 4 of the RAM.