Identifying Challenges to the Certification of Machine Learning for Safety Critical Systems

Eric Jenn[(1)], Alexandre Albore[(1)], Franck Mamalet[(1)], Grégory Flandin[(1)], Christophe Gabreau[(2)], Hervé Delseny[(2)], Adrien Gauffriau[(2)], Hugues Bonnin[(3)], Lucian Alecu[(3)], Jérémy Pirard [(3)], Baptiste Lefevre[(4)], Jean-Marc Gabriel[(5)],), Cyril Cappi[(6)], Laurent Gardès[(6)], Sylvaine Picard[(7)], Gilles Dulon[(7)], Brice Beltran[(8)], Jean-Christophe Bianic[(9)], Mathieu Damour[(9)], Kevin Delmas[(10)], Claire Pagetti[(10)]

[(1)] IRT Saint-Exupéry [(2)] AIRBUS SAS, [(3)] Continental, [(4)] Thales, [(5)] Renault Software Labs, [(6)] SNCF, [(7)] SAFRAN, [(8)] DGA, [(9)] SCALIAN, [(10)] ONERA

***Abstract*—Today, Machine Learning (ML) seems to be one of the only technically and economically viable solution to automate some complex tasks usually realized by humans, such as driving vehicles, recognizing voice, etc. However, these techniques come with new potential risks and as so, have only been applied in systems where the benefits of the technique are considered worth this increase of risk. But when dependability is at stake, the risk level must be contained. Giving confidence in the ML-based system to the developer of the system, to the regulation or certification authority that delivers the authorization to commission the system, or to the human operator that will interact with the system, becomes an essential objective.**

In order to identify the main challenges for placing a justifiable reliance on systems embedding ML and, eventually, certify those systems, the Institut de Recherche Technologique Saint-Exupery de Toulouse (IRT) has created a workgroup involving key players in the automotive, railways, and aeronautical domains.

This paper presents the objectives of this workgroup and the approach of the problem, and gives some first results. *Focus is placed on supervised machine learning.*

Keywords— Machine learning, certification, dependability

I. INTRODUCTION

If the introduction of Artificial Intelligence techniques in safety critical systems has been studied for quite a long time [1], considering the use of *Machine Learning* (ML) techniques in those systems is a only a very recent trend.

The successes of ML in solving difficult problems or improving significantly the performance of existing systems, make their dissemination ineluctable. But the challenges are as numerous as the opportunities.

In practice, ML techniques raise numerous challenges that could prevent them to be used in a system submitted to certification[1] constraints. But what are the actual challenges? Can they be overcome by selecting appropriate ML techniques, or by adopting new engineering or certification practices? These are some of the questions addressed by our Machine learning Certification Workgroup (WG).

In order to start answering those questions, our WG has decided to restrict the analysis to off-line supervised learning techniques, i.e., techniques where the learning phase completes before the system is commissioned. The actual spectrum of ML techniques is actually much larger, encompassing also unsupervised learning and reinforcement learning, but this seems to be a reasonable and pragmatic approach.

Our activities are part of the "Certification Mission" of the DEEL (DEpendable and Explainable Learning) project[2]. The WG is composed of specialists in the fields of certification, dependability, AI, and more generally, embedded systems development. Industrial and academics members represent the domains of aeronautics (Airbus, Apsys, Safran, Thales, Scalian, DGA, Onera), railway (SNCF), and automotive (Continental, Renault). Some members of our workgroup are involved in both the AVSI[3], the SOTIF [2] projects, and the new EUROCAE WG-114 (see III.A). DEEL industrial partners come from various industrial domains where trusting a system means, literally, accepting to place one's life in the "hand" of the system.

This workgroup is definitely not the only initiative in that direction, but it leverages on the favourable context in which it takes place, including the proximity of the members of the DEEL "core team" that gathers at the same location researchers in statistics, mathematics and AI coming from partner laboratories (IMT, IRIT, LAAS), and specialists and data scientists coming from the IRT and the industrial partners.

This paper presents the preliminary results of the workgroup. Work is still ongoing, but we consider that our analysis is already worth being shared, and discussed.

Towards that goal, the paper is organized as follows. Section II recalls the overall context, gives a brief overview of our workgroup, and presents other initiatives on this topic. Section III presents the main certification challenges and some new certification approaches that could facilitate the introduction of ML techniques in critical systems. Section IV describes the methods followed by the workgroup to identify the major problems for the adoption of ML in safety critical systems. Finally, Section V gives a synthesis of the challenges, proposes actions to address them, and concludes the paper.

II. MOTIVATION

A. Why using ML?

Before explaining why addressing the certification of embedded systems hosting ML techniques is necessary, it may be worth saying a few words about **why** we are considering integrating ML in those embedded systems.

[1] In this document, we use the term "certification" in broad sense which does not necessarily refer to an external authority.

[2] DEEL is a Franco-Canadian research project hosted, for the French part, at IRT Saint-Exupery. It is part of the overall ANITI (Artificial and Natural Intelligence Institute), one of the four 3IA institutes created by the French government in 2019. The workgroup also benefits from the support of the RTRA STAE (Réseau Thématique de Recherche Avancée pour l'Aéronautique et l'Espace) through the DAAVVE "tremplin" project."

[3] AVSI is the Aerospace Vehicle Systems Institute, see https://avsi.org.

Technically, ML solves problems that were generally considered intractable, such as the processing of natural language, the recognition of objects, and more generally the extraction of complex correlations in huge data sets and their exploitation in real-time. The spectrum of possible applications is huge, from giving autonomy to cars, aircrafts or UAVs, offering new man-machine interaction means, to predicting failures of systems, possibly leading to an improvement of the overall safety of systems. In addition, ML also provides better solutions to problems already solved using "traditional" algorithms. For instance, the ACAS Xu anti-collision system implemented with ML uses 1/1000 the amount of memory needed for a classical, table-based, implementation [3].

So, assuming that the use of ML is justified, compliance of these techniques with applicable safety and security constraints, and certification requirements remains to be discussed and, possibly, demonstrated.

B. The Machine Learning Certification Workgroup

The aim of the Machine Learning Certification workgroup is twofold: pointing out the major challenges that could prevent the adoption of ML techniques in our domains, and propose new approaches to address them. Towards that goal, our activities are organized along three main axes:

- Sharing knowledge between certification and ML communities
- Identifying the main difficulties raised by the usage of ML in safety critical systems
- Feeding the DEEL and IRT research teams with technical challenges.

1) Sharing knowledge, or acculturation

Standards like DO-178C, EN50128 or ISO 26262 have been elaborated carefully, incrementally, and consensually by software engineers. We consider that the same approach must apply to ML techniques: the emergence of new practices must be "endogenous", i.e., take roots in the ML domain, and be brought by data science and software engineers. So, in order to facilitate this emergence, one important activity of the workgroup is to share knowledge between certification specialists and Machine Learning experts to create a common understanding of the certification stakes and ML technical issues.

2) Identifying challenges for the acceptance of ML

Once a common ground has been established between ML and certification experts, the next step is to identify challenges considering conjointly (i) the characteristics of the ML techniques, and (ii) the certification constraints.

3) Feeding the research effort

The physical and organizational proximity of the "core team" of the DEEL project and the WG is a singularity compared to other initiatives. In practice, members of the core team, i.e., data scientists and mathematicians from industry and academia participate actively to the workgroup meetings. Those discussions feed AI and Embedded Systems researchers with accurate and sharp challenges directly targeted towards certification objectives. Some first results are given in Section V.

C. Other initiatives

Literature on the question of designing safe systems embedding ML algorithms is large and is growing exponentially. Among pertinent references, some address the general problem of AI, as in [4] or [5]. Other interesting references are more domain specific, such as [6] or [7] about autonomous automotive systems, or [8]–[12] in the aerospace domain.

Research projects and workgroups are popping-up all over the world in proportion to the enthusiasm, expectations, and socio-economic stakes raised by those techniques. However, we will simply mention three major initiatives to which some of the WG members participate: the AVSI AFE 87 project, "Certification Aspects of Machine Learning", the recently created EUROCAE WG-114 workgroup on Artificial Intelligence, and the SAE G-34 workgroup "Applied Artificial Intelligence in Safety-Critical Systems".

The AFE-87 project definitely shares objectives with our workgroup (see [13], page 46): "how to demonstrate a system incorporating ML meets its intended function in all foreseeable conditions", "how to demonstrate that a data set is correct and complete", etc. The objectives of the EUROCAE WG-114 are[4] similar: "to establish a comprehensive statement of concerns versus the demonstration of conformity of AI-based products to the airworthiness requirements", "clarify the scope of applicability" of AI techniques, and provide material "for developing AI technology embedded into and/or for use with aeronautical systems in both aerial vehicles and ground systems".

Note that none of these workgroups have published results yet. Moreover, those projects are focused on aeronautical domain whereas ours covers multiple industrial domains.

III. CERTIFICATION CHALLENGES AND NEW APPROACHES

Standards in the aeronautics domain (ARP4754, DO-178C, DO254, etc.), railway domains (EN50126, EN50128, EN 50129), or automotive domains (ISO26262) have been elaborated, and later updated, to take into account the evolution of technologies. In the aeronautical domain for instance, the DO-178 has been extended with technology specific supplementary documents that integrate new practices in software engineering, such as model-based driven engineering (DO-331), object-orientation (DO-332), and formal methods (DO-333). So, wouldn't it be possible to simply update or complement the existing standards to address ML techniques?

No. This approach seems hardly scalable as, the specificities of the problems addressed by ML and the new computation paradigms on which those techniques rely make them hardly compatible with the existing standards.

There is definitely a need for deeper changes, and those changes could leverage on the ongoing initiatives in the various industrial domains.

A. Certification challenges at a glance

Here, we briefly skim though four main problems that ML system developers are facing: (i) the specification problem, (ii) the dataset constitution problem, (iii) the learning problem, (iv) the implementation problem.

Concerning (i), ML techniques really excel in solving problems that are difficult to specify in the « traditional » way. Pedestrian detection is one classical example: even though ML algorithms perform pretty well at detecting pedestrians (in some conditions), defining what a pedestrian is remains an open issue. Then, how to become confident in a system, and how to achieve its certification, on the basis of a partial

[4] See https://www.eurocae.net/news/posts/2019/june/new-working-group-wg-114-artificial-intelligence/.

specification is a critical question, especially in application domains where the environment is highly variable and complex, and hardly controllable. We will see in the next section how this issue is addressed by the automotive domain via the SOTIF.

Concerning (ii), to demonstrate that the dataset is "sufficient" with respect to the intended function operating in the foreseeable operational conditions is definitely a certification challenge. Reciprocally, and this is what makes ML very valuable, the dataset can also capture important behaviours that would probably be missed by an explicit, human written, specification. This makes the validation activity even more complex.

Concerning (iii), either one is able to sufficiently verify the model produced by the training process with regard to the intended function or confidence on the learning process will have to be provided. Moreover, ML techniques are often opaque: they produce results hardly explainable, interpretable ([2], [3]) and then understandable. However, understandability is generally a prerequisite for trust. The lack of explainability makes it difficult to be confident in the capability of the model to generalize correctly on data different from the learning dataset, should they be in or out of the functional range.

Concerning (iv), ML *inference* algorithms are fairly "classical" for they basically involve standard linear algebra (for artificial neural networks) or decision structures (for decision trees). However, their implementation raises several issues that, if not strictly specific to ML, are here both amplified and concomitant. This includes the strong dependency on data, the importance of floating point computations – including floating point in parallel computations –, the high-level of parallelism, the reliance on accelerators such as GPU or dedicated accelerators, the use of complex tools and software libraries (e.g., Tensor Flow[16], Keras[17], etc.). Note that we focus here on architectures where inference is performed locally (embedded inference), excluding other solutions relying on remote computing devices (e.g., cloud) which would raise a new set of difficulties.

B. *Differences with current development standards and practices*

Once the learning phase is completed, the emerging behavior of a ML algorithm – i.e., its contribution to the *intended function* – essentially depends on *data*[5], such as weights, biases, and initial values in the case of artificial neural networks. This dependency of the behavior on data is not new: so-called "configuration data" in classical software usually play a similar role. However, the effects of those data can usually be translated into some effects on the program's control flow. This is not the case for weights and biases. Unfortunately, standards usually focus verification activities on code, as the main origin of the emerging behavior.

The same standards also recommend a requirement-based approach for making the demonstration of conformity. Software is specified, designed and coded following a traceable cascade of requirements from the needs (system requirements allocated to SW) to the code, and the object code for the most critical system. Then each software layer is verified against the upper level and the SW executable is verified against the whole set of requirements. On the contrary, a ML-based SW is neither developed based on requirements nor capable of having a design traceable to the functional needs. Therefore, a certification approach based on DO-178C or ISO 26262 standards is not immediately applicable to ML-based software. Some deep refactoring – possibly re-foundation – of existing standards seems necessary to take into account those data in proportion to their importance.

What precedes is obviously an extremely partial view of the dimension and complexity of the problem. To make it short, we believe that in order for the ML-based systems to become certifiable we need to address fundamental aspects both from the ML side and from the side of the certification methodologies. Now, let's have a look to the certification side.

C. *New "certification" practices in the automotive domain*

In the automotive domain, the Safety Of The Intended Function (SOTIF) workgroup addresses the problem another way, by considering that for the type of systems targeted by ML, an incomplete definition of the environmental conditions should not be considered exceptional, but "normal".

The SOTIF classifies operational scenarios according to their impact on the safety (safe/unsafe) and the *a priori* knowledge one may have concerning their occurrence in operation (known/unknown). This is represented on Figure 1.

As any other methodology aimed at ensuring safety, the main objectives of the SOTIF is to maximize or maintain the "safe areas", i.e., the proportion of operational scenarios leading to a safe situation, and minimize the "unsafe area", i.e., the proportion of operational scenarios leading to an unsafe situation, while keeping a reasonable level of availability of the system under design.

The originality of the SOTIF lies in the fact that it addresses explicitly the case of the *unknown* and *unsafe* operational scenarios. It proposes an iterative approach to reduce the occurrence of these situations "as much as possible with an acceptable level of effort".

The SOTIF is complementary to the ISO 26262. The ISO 26262 propose means to handle design errors (i.e., "malfunctions" of the system); the SOTIF extends the ISO 26262 by considering the effects of the "the inability of the function to correctly comprehend the situation and operate safely […including] functions that use machine learning", and the possible misuse of the system.

The specific limitations of the ML algorithms, including their capability "to handle possible scenarios, or non-deterministic behaviour", typically fits in the scope of the SOTIF.

The SOTIF proposes a process to handle those situations. In particular, it proposes an approach to verify the decision algorithms (See §10.3 of the SOTIF) and evaluate the residual risk. For instance, Annex C proposes a testing strategy of an Automatic Emergency Braking (AEB) system. An estimation of the probability of rear-end collision in the absence of AEB is obtained from field data.

Considering that the probability of occurrence of the hazardous event shall be smaller than this probability, the number of kilometre of data to be collected (for the different speed limits) in order to validate the SOTIF can then be estimated.

[5] Code is also data, obviously. If it were necessary for the comprehension, we could discriminate code and data as follows: code is what describe the operations to be performed on data. Translating a code into an operation is performed, for instance, by a processor.

The process may be iterated up to the point where the estimated residual risk is deemed acceptable.

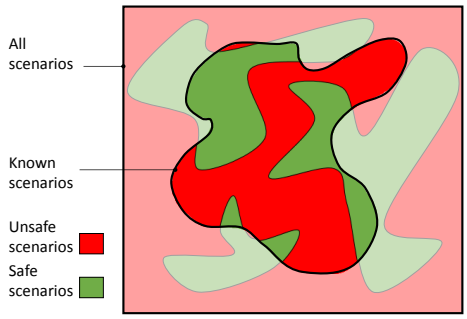

Figure 1. Operational scenarios classification as per SOTIF

This approach *allows but manages* the performance limitations inherent to ML. In particular, it explicitly allows unknown situations for which no learning data will be provided during the learning phase, but provides a methodological framework to handle them in a safe manner.

D. New certification practices in the aeronautics domain

In the aeronautic domain, a first initiative called *Overarching Properties* (OPs), has been explored by Airworthiness Authorities and industrials since 2015 in the frame of a global streamlining activity [18]. The initial objective was to come back to a necessary and sufficient set of fundamental properties (the "OPs") which, if possessed by a product, would ensure that it is suitable to be installed on an aircraft.

Three OPs have been identified: **intent** (the defined intended behavior is correct and complete with respect to the desired behavior), **correctness** (the implementation is correct with respect to its defined intended behavior, under foreseeable operating conditions), and **innocuity** (any part of the implementation that is not required by the defined intended behavior has no unacceptable impact).

Demonstrating that the product actually possesses those properties is the most difficult part. However, with respect to classical approaches driven by current standards such as the DO-178C, DO-254, and ARP4754A, OPs open the door to alternative approaches where compliance with the OPs is not demonstrated by showing that a list of pre-defined objectives is fulfilled, but by building an explicit argument [18].

Assurance cases (ACs) [19], [20] are *one* possibility to build and formalize this argument. ACs are already used in various industrial domains (e.g., [21]) and, in particular, have already been considered in aeronautics [22], [23], and automotive [24]. They have already been applied to Neural Networks [25]. We provide an example of an AC in Section IV.A.3.

The second alternate approach, named *abstraction layer* (AL) has been recently launched under responsibility of Airworthiness Authorities and a panel of Aerospace industrials. While targeting similar objectives as the OPs, it proceeds "bottom-up" by capturing the essence of existing, proven practices, into the abstraction layer.

OPs and AL are not dedicated to ML but they represent a re-foundation of the existing certification practices, and consequently, are an opportunity to build new certification methods and standards "adapted" to ML.

Finally, the WG-114 and G-34 workgroups (see Section II.C) have the objective to develop standards and/or guidance material for the certification/approval of aeronautical safety-related systems based on AI-technology. These groups are open to innovative practices, including but not limited to OPs and AL.

E. New "certification" practices in the railway domain

The railway domain regulation is already flexible: if a new specification and/or evaluation method needs to be developed to address ML solutions, the equipment manufacturer has to declare how it completes or deviates from the applicable Technical Interoperability Specification (TSI), and submits this declaration to the European Commission for analysis. The Commission can ask the European Railway Agency for its opinion on the proposed innovative solution. If the commission's opinion is favorable, then the new specification and evaluation method are developed and integrated into the

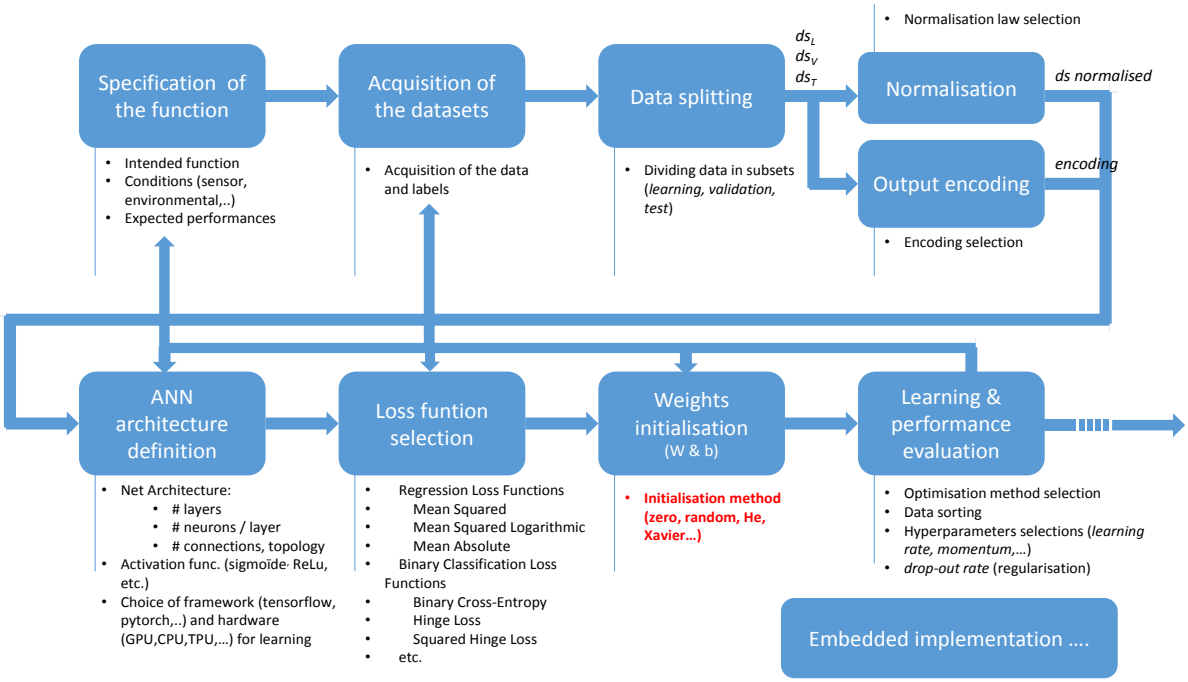

Figure 2. Simplified Machine Learning Workflow

Technical Interoperability Specification during the review process.

IV. SYSTEMATICAL IDENTIFICATION OF CHALLENGES

The ultimate aim of this work will be to point out the « challenges » for ML certification. Since certification is taken in a broad sense, we call *challenge* a situation that leads to a *deficit of confidence* on the capability of the ML system to perform its indented function. A deficit may be the consequence of the difficulty or impossibility to perform some usual V&V activities with the expected level of confidence (e.g., testing), or be the result of the introduction of new risks inherent to the method (e.g., the existence of a learning phase). A challenge may concern any phase and level of the development process, from specification to validation, at system, software, and hardware level.

To make the identification of challenges more systematic, we propose to do a detailed analysis of a typical ML-system development process (sec. IV.A) in order to identify where faults may be introduced and how they can be *prevented*, *identified* and *removed*. A difficulty in achieving any of these three actions is considered to lead to a deficit in confidence, and so to a potential "challenge". In a complementary manner, we also propose two approaches aimed at avoiding to face new challenges. The first one searches for similarities between ML techniques and techniques already implemented in certified systems (sec. IV.B). The second one searches for ML-techniques that do not show *some* of the other techniques' challenges (sec. IV.C). Those approaches are briefly introduced and illustrated hereafter. Their systematic application is an on-going process at the time of writing this paper.

A. ML development process analysis

The "process analysis" operates in a bottom-up manner: it determines the effect of the actions and decisions taken by a ML designer on the correctness of the service delivered by the system. The approach, similar to a software Failure Mode Effects Analysis (FMEA) is aimed at pointing out the locations in the process where faults having an impact on the final safety of the system are the most likely to be introduced.

Note that, by construction, the results produced by a ML system will generally be "erroneous" due to statistical nature of the learning process. A *fault* in the ML design process is a human-made action (including choices) that has an impact on the overall safety of the system.

The "process analysis" shall cover all engineering activities. For an implementation based on neural networks, for instance, it shall include the specifications and definition of the datasets, the choice of the initial values of the weights, the choice of the activation function or optimization method, the choice of the implementation framework, etc.

1) Workflow analysis

A partial[6] representation of a typical ML workflow based on neural networks is given Figure 2. This diagram shows the main phases along with some of the actions taken by the designer. For instance, during the "weight initialization phase", the designer has to select one particular method among those available. In this phase, a fault may be introduced by choosing an inappropriate method (for instance, using an "all-to-zero" initialization, see Figure 3). This example is obviously trivial but it illustrates the principle of the analysis. As in any FMEA, the objective is to identify the consequence on the system's behavior, and to define appropriate prevention, detection and elimination, and/or mitigation means. These means involve development process activities (i.e., provide guidance to prevent the fault to be introduced, provide appropriate verification means to activate and detect the error, etc.) and system design activities (to mitigate the effect of the error).

2) From engineering choices to High-Level Properties

Our experience shows that a particularity of ML techniques is that determining precisely the consequences of an engineering choice on the overall safety of the system is difficult. Therefore, as an intermediate step, we have proposed to define a set of High-Level Properties (HLPs) and to relate those properties to the ML engineering choices. A HLP is a property that, if possessed by a ML technique, is considered to have a positive impact on the capability to give confidence on the ML-system. So, a HLP is not a property of the system.

An excerpt of the complete list of HLPs is given in Table 1. Those definitions have been discussed among members, considering the literature and the standards. In some cases, such as for "Interpretability and Explainability", a pragmatic and relatively broad definition has been chosen to cope with the diversity and contradictory definitions found in the literature.

Once the HLPs defined, we propose to establish the causal relationship between them and the engineering choices. For instance, the HLP "accuracy" is related to some low level design choice via a chain of positive or negative effects. This is illustrated on Figure 3, which shows the path from the engineering choice (here, the choice of the initialization and activation functions) up to the high-level property. This systematic analysis (still in progress at the time of writing) is aimed at explaining the effects of ML design choices on confidence they can give on ML-system through the HLPs.

Table 1. High-level properties (excerpt)

HLP	*Definition*
Interpretability/ Explainability	Extent to which a ML system can provide an explanation about a decision in a form understandable by a human (adapted from [26])
Monitorability	Extent to which a system provides information that allows to discriminate a "correct behaviour" from an "incorrect behaviour"
Accuracy	Extent to which a ML system provides a result near to the true value
Data quality	Extent to which data are free of defects and possess desired features
Maintainability	Extend to which a system can be extended or improved without degrading its initial properties (no regression).
Auditability	Extend to which an independent examination of the development and verification process of the system can be performed
Robustness	Ability of the system to perform its intended function in the presence of: a) Abnormal inputs (e.g. sensor failure) [27], b) Unknown inputs (e.g. unspecified conditions)
Regularity	Extent to which the response of a system to an input I can be predicted from the knowledge of the response of the system to another input in the neighbourhood of I.
Provability	Extent to which mathematical guarantees can be provided that some functional or non-functional properties are satisfied [28].
…	

[6] For sake of brevity, no details are given in this paper about important parts of the process such as the "Learning and performance evaluation part"; some other phases are simply not represented (e.g., "implementation phase").

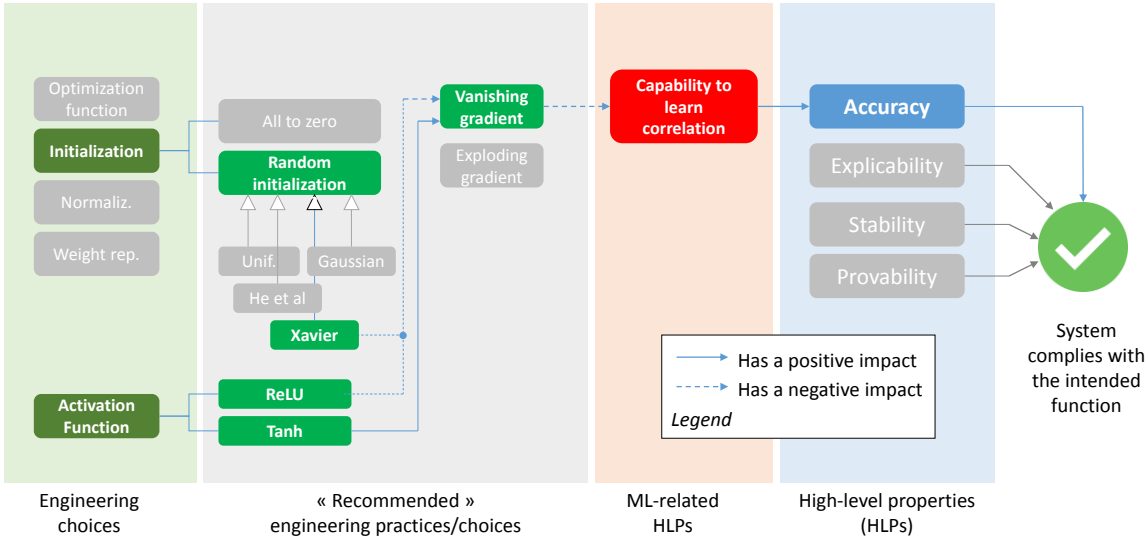

Figure 3. From High-Level Properties to Engineering choices

3) From HLP to assurance cases

As mentioned in Section III, future certification standards aim at facilitating the use of an explicit exposition of the reasoning using, for instance, Assurance Cases (ACs). But what is the relation between HLPs and ACs?

A HLPs is not a property of the system under design (SUD); it is a property of a method in the sense that "if the method owns the HLP then the demonstration of some property of the SUD will be facilitated". For instance, if a method owns HLP "provable", that means that it allows the use of mathematical proof to verify some property of the SUD. So, an AC is not aimed at demonstrating the possession of a given HLP, but it can be used to illustrate how possessing a given HLP contributes to building a convincing argument. Additionally, it illustrates how a convincing argumentation can be built without referring to a predefined set of objectives such as those found, for instance, in the DO-178C, Annex A.

Hereafter, we *illustrate* the approach by giving a short excerpt of a preliminary Assurance Case designed for a hypothetical automatic braking system implemented by a linear predictor. Here, focus is placed on the *provability* HLP, i.e., on the capability of the selected ML technique to ground the justification on mathematics.

The argument is organized as a tree of **claims**. Each claim is supported by a set of **premises** or by an **evidence**. The fact that the premises entails the claim is supported by a **justification**. A premise may raise other claims, recursively, or may be supported by an **evidence**, which is a leaf of the argumentation tree.

For instance, let us consider the part of the argument starting with the following claim (which is not the root claim):

ARGUMENT 0
CLAIM
In the foreseeable operational conditions, the system stops the vehicle before reaching an obstacle with $p_{fail} \leq 10^{-5}$
PREMISES
1. The foreseeable operational conditions are defined and the system is only allowed to operate in the defined operational conditions. If those conditions are not satisfied, control is given back to the driver with $p_{fe} \leq 10^{-6}$.
2. The braking distance[2] is estimated with $p_{fe2} \leq 10^{-6}$

[...]
JUSTIFICATION
By expert judgment, if the operational conditions are in the specified domain and the distance to the obstacle is not overestimated (including a non-detection of the obstacle), and the braking distance is not underestimated, and braking device behaves correctly, then the claim is established.
END ARGUMENT

Let's consider the second premise, which becomes claim #2:

ARGUMENT 2
CLAIM
The braking distance is estimated with $p_{fe} \leq 10^{-6}$
PREMISES
1. The speed is the only significant inputs to estimate the braking distance.
2. The braking distance is computed from the speed (v) of the vehicle using a linear Gaussian[1] prediction model with $p_{fe} \leq 10^{-6}$.

JUSTIFICATION
By expert judgment, if the speed is the only significant variable required to estimate the distance and the linear Gaussian prediction model is such that $p_{fe} \leq 10^{-6}$ then the braking distance can be estimated with $p_{fe} \leq 10^{-6}$.
END ARGUMENT

The second premise becomes a new claim (2.2) with its own set of premises:

ARGUMENT 2.2
CLAIM <see above>
PREMISES
1. The probability of underestimating the braking distance using a linear Gaussian prediction model under a fixed design approach [29] is smaller than 10^{-6}.
2. The linear Gaussian prediction model is a valid assumption.
3. The "fixed design approach" does not rely on any operational hypothesis

JUSTIFICATION
[...]
END ARGUMENT

The first premise of claim 2.2 represents a leaf of the evidence tree:

ARGUMENT 2.2.1
CLAIM <see 1st premises of claim 3.2>
PREMISES
1. The estimated braking distance for a given input x_{new} is calculated with the formula $\sum_{j=1}^{p} \hat{\beta}_j x_{\text{new,j}} + t_\delta \hat{\sigma} \sqrt{x_{new}^T (X^T X)^{-1} x_{new} + 1}$ where t_δ is the quantile of order $1-\delta/2$ of the t-distribution with n-p degrees of freedom
2. The estimated braking distance is higher Y_{new} with a probability of 1-δ

EVIDENCES

(Numerical application using the elements given in the justification.)

JUSTIFICATION

Well-known property of linear regression [30]: a mathematical proof of the prediction confidence interval on Y_{new} with probability $1-\delta$.

[…]

END ARGUMENT

For stake of conciseness, other parts of the argumentation tree are omitted in this paper, but this excerpt illustrates how we can estimate the *provability* of a ML technique (here a linear predictor) by building part of the argumentation tree and providing the required mathematical evidences. This exercise also points out how difficult it is to build a complete and convincing argumentation, even for a simple linear regression.

B. Similarity analysis

Another approach to analyze ML with respect to certification objectives is to search for situations where non-ML systems showing characteristics "similar" to those encountered on ML systems have been successfully certified. For the approach to be useful, similarity must concern peculiarities of either the ML techniques or of the problems that they address. One interesting example is the case of Kalman Filters (KF), a predictive filtering technique used, for instance, to compute the position and attitude of robots or aircrafts.

The KF algorithm shows similarities to ML techniques in the sense that (i) it implements a statistical estimation process, (ii) its behavior depends strongly on empirical data (the covariance matrix) and on hypotheses on the inputs (the distribution of the noise), (iii) it produces outputs that are associated with an estimation of the result quality (an error ellipsoid). Due to (ii), and as for ML, verifying the correctness of the algorithm *implementation* is not sufficient: a combination of mathematical demonstrations and tests is required to assess the intended function (including precision, stability, etc.). The use of KF in aircraft positioning systems is covered by the DO-229 standard [31] which gives very detailed recommendations about validation of such algorithms (type and number of tests), on the basis of a precise (yet statistical) knowledge of the system's environment (satellite positioning errors, gravity model, ionospheric error models, etc.). Hence, confidence is obtained by applying a function-specific standard.

Further work is still needed to determine to what extent such statistical approaches could be generalized and applied to certify ML-based systems. The computing power needed to reach statistical representativeness is less and less an issue, thanks to supercomputers and cloud computing. Therefore, safety demonstration based on massive testing could be a new way to be explored for ML certification.

Other similarities could possibly be found about verifying and validating activities of certified systems. For instance, the confidence about Worst-Case Execution Times estimations often relies on a combination of analytical and statistical arguments applied to a complex combination of hardware and software. However, the system is not considered as a black box. Statistical estimations are strongly supported by analyses of the hardware and software (see e.g., CAST-32 [32]) to account for the initial conditions of the components of the hardware platform, or subtle temporal conditions related to the presence of inter-core interferences, etc.

The question of explanability ("the extent to which the behavior of a machine learning model can be made understandable to humans") may also be addressed by analogy since it is a challenge for any complex software system (e.g. Cyber Physical Systems [33]) or complex hardware system. About hardware, for instance, when a COTS processor is used on an embedded system, transparency is only partial and, if ever it is, the hardware is usually so complex that it can essentially be seen as a black-box. So, how do current certification practices address this issue? And are those practices transferable to the problem of ML?

The development of complex hardware is covered by the DO-254 with some additional guidance given in [34]. For a COTS processor, the development assurance is essentially[7] ensured by the application of the DO-178C on the software: "The development assurance of […]highly complex COTS microcontrollers (Core Processing Unit) will be based on the application of ED-12B/DO-178B to the software they host, including testing of the software on the target microprocessor/microcontroller/highly complex COTS microcontroller". Stated differently, the hardware is considered to be tested along with the software it executes. This certainly raises questions when considering very complex microcontrollers. This issue is addressed in [35] which promote the idea of introducing mitigation means.

C. Backward analysis

The "backward analysis" takes the problem the other way round by considering *first* the solution possessing some of the HLPs, and *then* determine to which problems they can be applied, with what guarantees, and under what limits. We illustrate this approach with decision trees and NNs.

1) Decision trees and explainability

The property of explanability has been introduced in the previous section. A lot of activity is currently ongoing on that subject for the mainstream ML technologies such as DNN. Well known examples (e.g., classification of husky and wolves [36]) even show that those reasons may sometime be *very* arguable. However, explanations may be needed, by different actors and for different reasons: a pilot may need explanations to understand the decision taken by its ML-based co-pilot in order to be aware of the current situation and take control of the system if needed; an engineer may need explanations to investigate the origin of a problem observed in operation (e.g., to ensure the "continued airworthiness" in aeronautics [37]); etc. Explanations may be required to understand and then to act or react, but also to gain confidence on the system that takes decisions.

All ML techniques are not equivalent with respect to explanability. For instance, decision trees are "naturally" more explainable than DNN (even though some recent techniques tend to provide some explanability to DNN [38]).

A decision tree predicts the label associated with an input taking a series of decisions based on distinctive features [39]. Providing an explanation simply consists to expose the series of decision that were taken, from the root of the tree to the leave (the decision). This property enables to provide a full description of the decision process after learning, and a com-

[7] Airworthiness authorities actually requires more than testing (see Section 6 of the Notice of Proposed Amendment of the AMC-20 which introduces the AMC 20-152 on airborne electronic hardware), but it remains that the COTS is essentially a "black box" for the applicant.

prehensive set of Low Level Requirements for software implementation. Such Decision Trees could be used for different types of applications such as classification or regression problems [40]–[42].

2) ReluPlex and provability

Formal verification methods (FM) such as deductive methods, abstract interpretation, or model checking, aims at verifying that a model of a system satisfies some property on a mathematical basis. Being grounded on mathematics, these methods bring two fundamental properties, namely soundness and completeness, which can hardly be achieved by other means such as testing or reviews. FM are already considered as viable verification methods by authorities (see DO-333 supplement to the DO-178C).

In the current context, the objective is to demonstrate the compliance of an ML implementation to its specification in all possible situations, without explicitly testing the behaviour of the system in each of them. Completeness is very difficult to achieve by testing for problems with a very large dimensionality (as it is often the case with problems solved by ML). Whatever the testing effort, it will only cover an infinitesimal part of the actual input space. Confidence in the test could be improved drastically if equivalence classes can be defined. An equivalence class is defined with respect to some fault model so that "any test vectors in a class with discover the same fault". Unfortunately, it is not yet very clear how those classes can be defined in the case of, for instance, NNs. Being able to apply a formal verification technique on a ML design would represent a significant improvement over testing.

So, are there ML techniques amenable to formal verification? The answer is yes: formal methods have already been considered in the domain of Machine Learning [43].

Here, we consider the Reluplex method proposed in [3]. This method is used to verify properties on NN with ReLu activation functions. It is based on the classical Simplex algorithms extended to address ReLu constraints. As for the Simplex itself, the technique gives satisfying results in practice despite the problem being NP-complex. The ReluPlex method is applicable to NN using ReLu activation function; it has been applied successfully on NN of moderate size (300 nodes, fully connected).

ReluPlex has been used on NN-based implementation of an unmanned collision avoidance system (ACAS-Xu), a system generating horizontal manoeuvre advisories in order to prevent collisions. The ML implementation of the ACAS Xu is a small memory-footprint (3Mb) alternative to the existing implementation based on lookup tables (2Gb)[8].

This technique provides formal guarantees on properties that can be expressed using linear combinations, which means that only a subset of the possible functional properties can be verified that way. In practice, the formal verification effort has been focused on safety properties such as "no unnecessary turning advisories", "strong alert does not occur when intruder vertically distant", etc. Reluplex has also been applied to demonstrate the robustness property, such as the δ-local and global robustness[9]. Such formal proof could be used as a mean of compliance with the specifications, even without taking into account the learning phase.

V. CHALLENGES, ACTIONS, AND CONCLUSION

A challenge is a situation leading to a deficit of confidence on the capability of the ML system to perform its indented function. With the introduction of the HLPs (§IV.A.2), a challenge is now a difficulty to satisfy an HLP.

A first list of challenges has been established by the working group. They are presented hereafter according to four main axes: *fault avoidance*, i.e., how to prevent the introduction of faults in the system; *fault*[10] *removal*, i.e., how to eliminate faults that have not been avoided; *fault tolerance*, i.e., how to cope with faults that have not been removed, and *new certification practices*, i.e., how to give confidence on the system.

1) Fault avoidance

The best way to cope with faults is to avoid their introduction, at all levels of the development process, from the specification to its implementation.

Specifying a ML is the very first difficulty, especially in situations where the function to be performed can hardly be specified rigorously and completely. In [45], for instance, the authors propose a "partial behavioral specification; other suggest to use the dataset as a specification. In both cases, estimating what the "specification" actually captures, and reducing the residual uncertainty to an acceptable level are two challenges. This is an aspect addressed by the SOTIF.

Considering the overall ML system development process, we can distinguish two main phases: the *learning phase* during which a ML model (NN, decision trees, etc.) is chosen and "educated", and the *inference phase* where the configured model is implemented on the embedded system. Each of these phases follows its own development cycle, with its own specification, development and V&V sub-phases. In theory, demonstrating the compliance of the overall system to its intended function could be done end-to-end without considering each phase separately, but in practice, the demonstration effort is distributed over the different phases.

The learning phase is definitely the most complicated part of the problem as it involves new engineering practices, at least in our domains. How to process and manage data is already covered by standards, but only for very specific data and purposes. In the case of ML, data (opposed to code) are no more simply processed by the system, or used a means to configure the system in a limited range, it completely determines the behaviour of the system. So, ensuring the quality of the data (correctness, representativity, etc.) is a crucial issue. Whether this issue shall be addressed by a specific "learning assurance" process [46], comparable and complementary to the classical "development assurance" process, or by a more flexible approach based on "assurance cases", is an open question.

ML is a statistical method. As illustrated on the simple case of the linear predictor (§IV.A.3), confidence on the results computed by ML depends on complex hypotheses about the input variables (distribution, existence of hidden variables, correlation, etc.). A rigorous definition of those hypotheses and a rigorous verification of the compliance with these hypotheses, is necessary to be able to place a justified confidence on the results produced by a ML system. Our exercise about the linear predictor is a first step towards making all

[8] The ACAS Xu was too complex for manual implementation. The strategies considered to reduce the size of the data required to implement the functions are described in [44]

[9] A DNN is δ-locally robust if for two inputs x and x' such that $\|x - x'\|_\infty \leq \delta$ the network assigns the same label to x and x'.

[10] In what follows, we only consider human-made faults, i.e. faults introduced by human during the different phases of the ML development.

hypotheses explicit; we expect this approach to be applied on real-size problems.

The engineering of the algorithm (selection of the activation functions, choice of the NN architecture, etc.) is also very specific and, as it offers many design alternatives, a large and open door to a whole new class of faults.

In software engineering, many standards have been published to propose good coding practices. We consider that building a "learning standard" or, at least provide some guidance on that matter, would be useful for the community. These studies will be done in accordance with DEEL core team works on mathematical guaranties for ML, biases detections in datasets and treatment, model robustness, etc.

Concerning the inference phase, it seems to raise only few *specific* issues. First, it is worth noting that ML techniques (e.g., DNN) propose generic models of computation supported by *generic model of implementations*. This opens the door to reusable qualified implementations of DNN, or qualified code generators that would significantly reduce the certification effort, for that part of the process. However; several difficulties remain to be addressed. First, the success of ML techniques strongly relies on the existence of a large – and ever growing – set of technologies, frameworks, libraries, code generators, etc. The technical debt about those technologies is increasing at an extremely high pace. The question of qualifying those elements against the certification standards or developing qualifiable version of those elements becomes critical. A collaborative effort to build the certification artifacts could be wise.

Besides, floating point precision and parallel computation are well-known difficulties that will be exacerbated by the large amount of floating point calculations required by ML techniques.

In the same way that an error correcting code detect or correct some specific physical fault model, to avoid, remove, or tolerate faults in an ML system, fault models must be built. This is one objective of the process analysis proposed in Section IV.A.1 which is aimed at identifying where and when faults may be introduced in the process. This systematic analysis has been initiated by the workgroup and will be pursued and completed.

2) Fault removal

Fault removal traditionally rely on four main methods: Inspection, Analysis, Demonstration, and Test (IADT), most of which raising specific difficulties for ML.

Analysis, for instance, requires building an abstraction of the solution that is used to check compliance with the expected properties of the system. ML techniques do not provide such model. *Test* is aimed at activating faults and propagating them to the system's output. The difficulty to specify the intended function, the absence of fault models, the absence of structural model upon which building coverage metrics, and the (usually) very large dimensionality of the input domain are all new challenges to be addressed. In addition, tests are also aimed at demonstrating the absence of unintended function. Then again, the absence of structural coverage criterion makes this demonstration much harder than for classical software. New testing approaches, metrics, and techniques – possibly leveraging on the computational capabilities available today (cloud) – are to be devised.

Alternate solutions to test must also be investigated. Formal methods such as ReluPlex (see IV.C.2) are possible solutions to complement or replace testing, but the applicability and limits of these methods have to be analyzed, as well as new formal methods investigated by the core team.

3) Fault tolerance

The numerous challenges identified about fault avoidance and fault removal make it clear that the safety of ML systems will initially rely on a combination of learning and development assurance practices **and** fault tolerance mechanisms based on error detection and various levels and forms of redundancy.

Here again, ML raise new challenges. Indeed, besides cases where safety properties can be formulated and captured in a monitoring device implemented using classical approaches, there are cases where there the only redundant implementation would also rely on ML, leading to the risk of common mode errors. Investigations are needed to determine ML systems can be monitored, using (when required) which other ML technique, with level of diversification and, at the end, which risk for common mode errors.

4) Towards new certification practices

Regarding software, the current certification framework is strongly adapted to "classical", procedural, forms of computations. Existing recommended development and V&V practices are still applicable on part of the process (see above), but they hardly give assurance about the implementation of the intended function itself. As already stated, the latter depends strongly on data, an element that is only marginally considered [11] in the current certification corpus. What is needed is recommendation and guidance about the *learning process*.

But the major difficulty for certification *practices* is to tackle the inherent uncertainty of machine learning. Up to now, the objective – or the *credo?* – has been that a correct software cannot produce an erroneous result. So, to ensure the correctness of the result with an acceptable confidence level, one has to produce a correct software, i.e., to prevent or remove design faults, and this was deemed feasible by applying an ever-growing corpus of certification objectives. Probabilistic software assessment (cf. all the works about software reliability) has never been considered as an acceptable means to gain confidence, but the case for ML is different since uncertainty *is* the rule.

B. Conclusion

In this paper, we have briefly presented the objectives of the ML Certification Workgroup set up at IRT in the context of the ANITI and DEEL projects. We have presented the issues raised by ML with respect to the current certification practices, and have shown how we could leverage on the evolution of the existing standards to address these issues. Then, we have proposed a combination of analyses to identify in a systematic way the challenges for the application of ML in safety critical systems. Finally, we have shown how these challenges become concrete scientific objectives for the *core team* of the DEEL project.

This work is still an on-going process. Results will eventually be synthesized in a White Paper to be published.

[11] Data are essentially considered as inputs for the algorithms. No dedicated verification activities are proposed, except for Parameter Data Item (cf. DO-188 C, §2.5.1). The verification of these data is described in DO-178C, §6.6. except under specific conditions, the verification of data is a by-product of the code verification. By if there is a test completeness criterion for the code, there is no such criterion for data…

ACKNOWLEDGEMENTS

The authors want to thank the funding members of the DEEL project, the ANR, the ANITI, the STAE RTRA for their support. We also thank M. Malenfant for his insightful comments.

REFERENCES

[1] L. H. Harrison, P. J. Saunders, and P. J. Saraceni, 'Artificial intelligence and expert systems for avionics', in *Digital Avionics Systems Conference, 1993. 12th DASC., AIAA/IEEE*, 1993, pp. 167–172.

[2] 'ISO/PAS 21448 - SOTIF', ISO/PAS 21448, Jan. 2019.

[3] G. Katz, C. Barrett, D. Dill, K. Julian, and M. Kochenderfer, 'Reluplex: An Efficient SMT Solver for Verifying Deep Neural Networks', *ArXiv170201135 Cs*, Feb. 2017.

[4] D. Amodei, C. Olah, J. Steinhardt, P. Christiano, J. Schulman, and D. Mané, 'Concrete problems in AI safety', *ArXiv Prepr. ArXiv160606565*, 2016.

[5] C.-H. Cheng *et al.*, 'Neural Networks for Safety-Critical Applications - Challenges, Experiments and Perspectives', *ArXiv170900911 Cs*, Sep. 2017.

[6] F. Falcini and G. Lami, 'Challenges in Certification of Autonomous Driving Systems', in *2017 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*, Toulouse, 2017, pp. 286–293.

[7] M. Wood, P. Robbel, M. Maass, R. Duintjer Tebbens, M. Harb, and J. Reach, 'Safety First for Automated Driving', 2019.

[8] C. Tschan, A. Yucel, and N. Nguyen, 'Roadmap for Intelligent Systems in Aerospace', AIAA - Intelligent Systems Technical Commitee (ISTC), Roadmap, Jun. 2016.

[9] S. Bhattacharyya, D. Cofer, D. Musliner, J. Mueller, and E. Engstrom, 'Certification considerations for adaptive systems', 2015, pp. 270–279.

[10] E. Alves, D. Bhatt, B. Hall, K. Driscoll, A. Murugesan, and J. Rushby, 'Considerations in Assuring Safety of Increasingly Autonomous systems', NASA/CR-2018-220080, May 2018.

[11] K. Woodham, 'Verification of Adaptive Systems', 2016.

[12] AIAA, 'Roadmap for Intelligent Systems in Aerospace', 2016.

[13] '2019 EUROCAE Symposium & 56th General Assembly', Toulouse, 25-Apr-2019.

[14] F. Doshi-Velez and B. Kim, 'Towards A Rigorous Science of Interpretable Machine Learning', *ArXiv170208608 Cs Stat*, Feb. 2017.

[15] F. K. Dosilovic, M. Brcic, and N. Hlupic, 'Explainable artificial intelligence: A survey', in *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, Opatija, 2018, pp. 0210–0215.

[16] Martín Abadi *et al.*, *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. 2015.

[17] F. Chollet and others, *Keras*. GitHub, 2015.

[18] C. M. Holloway, 'Understanding the Overarching Properties: First Steps', p. 36, 2018.

[19] S. E. Toulmin, *The Use of Argument*. Cambridge University Press, 2003.

[20] J. Rushby, X. Xu, M. Rangarajan, and T. L. Weaver, 'Understanding and evaluating assurance cases', 2015.

[21] C. Duffau, T. Polacsek, and M. Blay-Fornarino, 'Support of Justification Elicitation: Two Industrial Reports', in *Advanced Information Systems Engineering*, vol. 10816, J. Krogstie and H. A. Reijers, Eds. Cham: Springer International Publishing, 2018, pp. 71–86.

[22] T. Polacsek, S. Sharma, C. Cuiller, and V. Tuloup, 'The need of diagrams based on Toulmin schema application: an aeronautical case study', *EURO J. Decis. Process.*, vol. 6, no. 3, pp. 257–282, Nov. 2018.

[23] C. M. Holloway, 'Explicate '78: Uncovering the Implicit Assurance Case in DO-178C.

[24] A. Rudolph, S. Voget, and J. Mottok, 'A Consistent Safety Case Argumentation for Artificial Intelligence in Safety Related Automotive Systems', in *Proceeding of the ERTS 2018*, Toulouse, 2018.

[25] Z. Kurd, T. Kelly, and J. Austin, 'Safety Criteria and Safety Lifecycle for Artificial Neural Networks', p. 11, 2008.

[26] C. Molnar, *Interpretable Machine Learning: A Guide for Making Black Box Models Explainable*. 2019.

[27] RTCA, 'DO-178C Norm'. 2011.

[28] 'DO-333 - Formal Methods Supplement to DO-178C and DO-278A', RTCA, Inc., Standard DO-333.

[29] P. Rigollet, '18.S997: High Dimensional Statistics', 2015.

[30] Wikistat, *Régression linéaire simple — Wikistat*. 2016.

[31] 'Minimum Operational Performance Standards for GPS/WAAS Airborne Equipment (DO-229)', RTCA, Inc., DO-229.

[32] 'Multi-core Processors', Certification Authorities Software Team, CAST-32A, Dec. 2016.

[33] J. Greenyer, M. Lochau, and T. Vogel, 'Explainable Software for Cyber-Physical Systems (ES4CPS)', presented at the GI Dagstuhl Seminar 19023, 2019.

[34] 'Certificartion Memorandum - Development Assurance of Airborne Electropnic Hardware', EASA, CM-SWCEH-001, is. 01, rev. 02.

[35] P. Bieth and V. Brindejonc, 'COST-AEH - Use of Complex COTS in Airborne Electronic Hardware - Failure Mode and Mitigation', EASA, CCC/12/001303 rev. 05.

[36] M. T. Ribeiro, S. Singh, and C. Guestrin, '"Why Should I Trust You?": Explaining the Predictions of Any Classifier', 2016, pp. 1135–1144.

[37] 'Certification Maintenance Requirments', FAA, AC 25-19A, Mar. 2011.

[38] L. H. Gilpin, D. Bau, B. Z. Yuan, A. Bajwa, M. Specter, and L. Kagal, 'Explaining Explanations: An Overview of Interpretability of Machine Learning', *ArXiv180600069 Cs Stat*, May 2018.

[39] S. Shalev-Shwartz and S. Ben-David, *Understanding Machine Learning: From Theory to Algorithms*. Cambridge: Cambridge University Press, 2014.

[40] L. Spirkovska, D. Iverson, S. Poll, and A. Pryor, 'Inductive Learning Approaches for Improving Pilot Awareness of Aircraft Faults', in *Infotech@Aerospace*, Arlington, Virginia, 2005.

[41] A. B. A. Christopher and S. A. alias Balamurugan, 'Prediction of warning level in aircraft accidents using data mining techniques', 2014.

[42] N. E. I. Karabadji, H. Seridi, I. Khelf, and L. Laouar, 'Decision Tree Selection in an Industrial Machine Fault Diagnostics', in *Model and Data Engineering*, vol. 7602, A. Abelló, L. Bellatreche, and B. Benatallah, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 129–140.

[43] S. A. Seshia, D. Sadigh, and S. S. Sastry, 'Towards verified artificial intelligence', *ArXiv Prepr. ArXiv160608514*, 2016.

[44] K. D. Julian, J. Lopez, J. S. Brush, M. P. Owen, and M. J. Kochenderfer, 'Policy compression for aircraft collision avoidance systems', in *2016 IEEE/AIAA 35th Digital Avionics Systems Conference (DASC)*, Sacramento, CA, USA, 2016, pp. 1–10.

[45] R. Salay and K. Czarnecki, 'Using Machine Learning Safely in Automotive Software: An Assessment and Adaption of Software Process Requirements in ISO 26262', p. 56.

[46] 'The European Plan for Aviation Safety EPAS 2019-2023', EASA, Nov. 2018.