This pre-print is an expanded version of the peer-reviewed paper:
G. Pai, "Deriving Safety-related Performance Requirements for Machine Learnt Aeronautical Applications", in
Proceedings of the 44th AIAA DATC/IEEE Digital Avionics Systems Conference (DASC 2025), Montreal, Canada.

Relating System Safety and Machine Learnt Model Performance*

Ganesh J. Pai
KBR / NASA Ames Research Center
Moffett Field, CA 94035, USA
ganesh.pai@nasa.gov

Abstract

The prediction quality of machine learnt models and the functionality they ultimately enable (e.g., object detection), is typically evaluated using a variety of quantitative metrics that are specified in the associated model performance requirements. When integrating such models into aeronautical applications, a top-down safety assessment process must influence both the model performance metrics selected, and their acceptable range of values. Often, however, the relationship of system safety objectives to model performance requirements and the associated metrics is unclear. Using an example of an aircraft emergency braking system containing a machine learnt component (MLC) responsible for object detection and alerting, this paper first describes a simple abstraction of the required MLC behavior. Then, based on that abstraction, an initial method is given to derive the minimum safety-related performance requirements, the associated metrics, and their targets for the both MLC and its underlying deep neural network, such that they meet the quantitative safety objectives obtained from the safety assessment process. We give rationale as to why the proposed method should be considered valid, also clarifying the assumptions made, the constraints on applicability, and the implications for verification.

1 Introduction

Amongst the core outcomes of the safety assessment process for civil aircraft [1] are *quantitative safety objectives* (QSOs). They represent an acceptable upper limit on the average probability of events that result in adverse safety effects. As part of aircraft system development, QSOs are allocated across the system hierarchy, from aircraft functions to the implementing *items*. In conventional systems not including machine learning (ML), the decomposition, allocation, refinement, and eventual verification of QSOs has only been applied to hardware items.

QSOs are not considered for software and the programmable aspects of hardware, in part, because the prevailing assurance guidelines [2], [3] intentionally avoid concepts of quantitative reliability or failure probability. Instead the focus is on applying process rigor to identify and correct development errors. The goal is providing assurance to an adequate level of confidence that the implementations of software or hardware designs are correct. The extent of the necessary development process rigor, given in terms of *development assurance levels* (DALs), is proportional to the severity of the undesired effects identified from the safety assessment process.

Although QSOs and DALs each address a different type of concern, they are associated through the severity of the effects of function (or item) failures. In particular, functions (or items) whose failures lead to higher severity effects are assigned a proportionally higher DAL and lower QSO, than those causing lower severity effects. Moreover, safety verification expects to confirm that those functions or items have been developed to the assigned level of rigor, and also that, for the related hardware, the QSOs have been attained.

When integrating machine learning (ML)-based functionality into aircraft systems, it is anticipated that in addition to the assignment of DALs, allocating QSOs to *machine learnt components* (MLCs), and relating the corresponding

*This work was authored by an employee of KBR, Inc., under Contract No. 80ARC020D0010 with the National Aeronautics and Space Administration. The United States Government retains and the publisher, by accepting the article for publication, acknowledges that the United States Government retains a non-exclusive, paid-up, irrevocable, worldwide license to reproduce, prepare derivative works, distribute copies to the public, and perform publicly and display publicly, or allow others to do so, for United States Government purposes. All other rights are reserved by the copyright owner.

targets to the associated performance requirements and metrics will be mandated.[1] Performance metrics for MLCs can be seen in part as quantitative criteria giving a long-term characterization (i.e., over the duration of their intended use) of their behavior relative to their requirements. An MLC that fails to meet its requirements may lead to functional failures and thereby to system-level safety effects. As such, relating the performance metrics of an MLC to higher-level safety objectives facilitates capturing how it contributes to both safety and the overall functional intent.

Currently available guidance for integrating ML into aeronautical systems, e.g., [4], does not clarify how valid MLC performance requirements should follow from an allocated QSO. Nor does it clarify how safety-related metrics may be selected, which metrics may be invoked in those requirements, or what range of values may be admissible for those metrics. Although those questions have been previously identified and investigated in other safety-critical domains (see Section 7.1 for related work), so far as we are aware they have not yet been adequately answered. To that end, we adapt the *aircraft emergency braking system* (AEBS) from prior literature [5] as an illustrative example (described in Section 3), to make the following contributions in this paper:

- We describe an initial method to translate QSOs obtained from a safety assessment into the safety-related performance requirements and associated metrics for an MLC (Section 4).
- We develop an abstraction of the required behavior (Section 5) that: (i) traces to and meets the allocated QSO, and (ii) is suitable for determining safety-related performance metrics and parameters such as the required confirmation threshold for detecting an object of interest in an image sequence, a tolerable miss ratio for not detecting that object, and the per image probability of non detection—a metric directly linked to the generalization capability of the machine learnt model.

We supply the rationale to substantiate why our method and the resulting MLC performance requirements should be considered to be valid in Section 6. This section also discusses the additional considerations that result from the method, the constraints that apply, and the implications for verification.

2 Conceptual Background

This section introduces the concepts relevant for the rest of the paper.

2.1 Quantitative Safety Objectives (QSOs)

As mentioned earlier in Section 1, a QSO is an acceptable upper limit on the average per flight probability of an adverse event, usually normalized by exposure (itself expressed as a duration or a count). For systems and equipment installed on aircraft, QSOs may also be viewed as targets for reliability or, equivalently, probability of (the effects of) so-called *failure conditions*: aircraft-level conditions that can directly or indirectly affect an aircraft and its occupants, including the crew, caused by one or more system failures, in combination with operational or environmental conditions encountered during various flight phases.

Note that failure conditions are synonymous with *hazards*, as used in other safety-critical domains. Also, failures include both *loss of function* and *malfunction*, whose respective causes encompass but are not limited to one or more component failures and their combinations, common causes, unintended or undesired emergent system interactions, and development errors (including errors in requirements) and their respective effects.

QSOs are defined and selected such that they are inversely proportional to the severity of the credible worst-case safety effects identified in a functional hazard assessment (FHA). The acceptable range of values for a QSO relative to effect severity is codified in civil aviation regulatory guidance documents. For example, a failure condition of MINOR severity, characterized as resulting in “a slight reduction of functional capabilities or safety margins of the airplane, physical discomfort for passengers, or a slight increase in workload for the crew”, is associated with an allowable quantitative probability, i.e., a QSO, of 10^{-3} per flight hour (pfh) [6].

The decomposition and allocation of a QSO in aircraft system development follows the preliminary system safety assessment (PSSA) process [1]. That process contributes to a systematic evaluation of a system architecture to determine how failures of the architectural components lead to the failure conditions identified in the higher level FHA. The PSSA can employ different analysis techniques, such as fault tree analysis (FTA), and Markov models.

[1] Although aviation industry consensus-based guidelines for development and assurance of aeronautical systems integrating ML are still being crafted, some regulatory publications expected to inform aviation rulemaking have proposed to relate QSOs to MLC and MLM performance metrics [4].

A quantitative, combinatorial FTA serves to validate the failure probability (or reliability) budgets established for architectural components, by confirming that they lead to a probability of an identified failure condition that is no worse than the associated QSO. Such a validation is a bottom-up assessment. A top-down analysis may also be performed to decompose and allocate the QSO to the architectural components by leveraging the fault tree logic and various heuristics.

2.2 Machine Learnt Models and Components and Their Characteristics

2.2.1 Machine Learnt Model (MLM)

An MLM is a mathematical formula or mapping rule, $f : X \rightarrow Y$, constructed by applying learning algorithms to (training) data, which comprises examples of the (patterns of) behavior to be learnt [7]. Here, X is the input space (or domain, or feature space), and Y is the output space (or codomain, or space of responses). A deep neural network (DNN) is one possible such MLM. A description of X as captured in the MLM requirements is known as an *operational design domain* (ODD) [7].

2.2.2 Machine Learnt Component (MLC)

In this paper, an MLC groups hardware and software implementations of one or more MLMs and, when appropriate, the supporting functionality (such as pre- and post-processing) necessary for their execution. An MLC is treated as a single entity allocated a DAL and a QSO from a system standpoint.

2.2.3 Deterministic Behavior

A trained MLM that does not continue to learn in use is *static*. That is, once f has been constructed, it does not change given some future input $\mathbf{j} \in X$. As such, f is *deterministic* in the sense that, given a specific input (vector) $\mathbf{x} \in X$ for which the model produces a response (vector) $\mathbf{y} \in Y$, any future input $\mathbf{j} \in X$ that is identical to the input $\mathbf{x}$ will always produce the same response $\mathbf{y}$.

2.2.4 Systematic Behavior and Correctness

A suitable MLM is one that *generalizes* from the training data inputs to unseen inputs from X, producing the required responses from Y.

The response $\mathbf{y}$ for the input $\mathbf{x}$ is *correct* when it is the required response, otherwise it is *incorrect*. More generally, because f is deterministic, the responses of a static MLM to its inputs are *systematic* in being correct or incorrect. That is, the input $\mathbf{x}$ supplied at any future time point will always produce the same correct or incorrect response $\mathbf{y}$. Moreover, if $g : X \rightarrow Y$ is the *true* (but usually unknown) function relating the input and output spaces, then f is correct when for all $x \in X$, $f(x) = g(x)$. That is, f produces the correct response for any input from X, and is said to generalize perfectly.

However, uncertainties in various aspects of the ML process, e.g., epistemic uncertainty due to insufficient knowledge about the nature of g and, therefore, a suitable form for f, as well as aleatoric uncertainty when sampling from X, together sampling limitations, can often result in an f that may not always produce the correct responses for some subset of previously unseen inputs from X. Such imperfect generalization can be characterized in terms of the *generalization error*, a (performance) metric of how MLM responses in use differ or deviate from the required responses for previously unseen inputs. The generalization error cannot be exactly calculated, but instead, theoretically, it can be probabilistically bounded to give a *probably approximately correct* MLM [8], especially in the context of supervised learning (also see Section 5.3).

2.2.5 Failure Probability and Insufficient Performance

The inputs from X may be governed in general by some (possibly unknown) generating process. The individual inputs can then be described in terms of (empirical estimates of) their limiting relative frequencies and, in turn, as a probability function $\Pr_X(\mathbf{x})$. In fact, a careful characterization of $\Pr_X(\mathbf{x})$ is a key requirement when defining the ODD [9]. Given the preceding discussion (Sections 2.2.3 and 2.2.4), and assuming that f is not a constant function (i.e., f produces the same response for any input), when the inputs occur according to $\Pr_X(\mathbf{x})$, the relative frequencies of the responses can also be established. In other words, the responses can be described through a probability function, $\Pr_Y(\mathbf{y})$.

Now, for a discrete input $\mathbf{x} \in X$ occurring with a probability $\Pr_X(\mathbf{x})$, let $\mathbf{y} \in Y$ be the correct response, and let $\mathbf{1}_f(\mathbf{x})$ be an *indicator function* defined such that $\mathbf{1}_f(\mathbf{x}) \equiv 1$ when f returns an incorrect response (i.e., $f(\mathbf{x}) \neq \mathbf{y}$), and is 0 otherwise. Then, treating all incorrect responses as *failures*, we can define a *probability of failure* of an MLM as in (1), i.e., the limiting relative frequency of incorrect responses for an infinite sequence of random discrete[2] inputs $\mathbf{x}$ that occur according to the input space probability mass function $\Pr_X(\mathbf{x})$:

$$\Pr(f(\mathbf{x}) \neq \mathbf{y}) \overset{\text{def}}{=} \sum_{\mathbf{x} \in X} \mathbf{1}_f(\mathbf{x}) \Pr_X(\mathbf{x}) \tag{1}$$

Later (Section 5.3), we describe how such long-term failure behavior characterizes *insufficient generalization performance* of an MLM.

2.3 Performance Metrics and Requirements

Once an MLM has been constructed, quantitative metrics are typically used to evaluate its prediction quality, i.e., how well its responses to inputs not previously seen during its training and development, correspond to the functional intent and the required responses. Examples of some commonly used metrics include: (for classification problems) *precision*, *recall*, and *F1 score*, as well as (for regression problems) *mean absolute error*, and *mean squared error*.

When the type of the response of an MLM and its containing MLC are the same, then the same set of metrics may be used for each. For instance, an MLC classifying its inputs using an ensemble of classifiers can be evaluated using the same classification performance metrics as those used for evaluating the individual MLMs in the ensemble. However, the specific values of those metrics for each MLM in the ensemble may differ from the values of the same metrics when applied to the containing MLC.

For conventional aircraft systems not integrating ML, performance requirements describe specific attributes of functions or systems, such as the type of performance, accuracy, range, fidelity, resolution, and timing behavior [13].

In addition to the above, MLC and MLM performance requirements express the respective desired long-term behaviors, for which a probabilistic formulation may often be appropriate. More generally, they invoke the associated performance metrics and specify their admissible values.

Safety-related performance requirements for an MLC and MLM are those traceable to QSOs, or to higher-level safety requirements, whose violation causes or contributes to a failure condition of the containing (system or aircraft-level) function. Non safety-related performance requirements are, equivalently, those whose violation does not cause or contribute to failure conditions. This paper focuses primarily on the former.

3 Illustrative Example

To explain the derivation of safety-related MLC and MLM performance requirements and metrics from an allocated QSO, we consider an illustrative example as shown in Fig. 1—an aircraft emergency braking system (AEBS) adapted from the prior literature [5] as follows: unlike in Fig. 1, the architecture in [5] does not include pre-processing. Also, it treats the post-processing as a part of the emergency braking controller (EBC) functionality, and (implicitly) equates the machine learnt sign detector (MLSD) with the MLC.

This section summarizes only those aspects of the AEBS and its safety assessment that are a necessary background for this paper. For more and other details on the AEBS, we refer the reader to [5].

3.1 System Description

3.1.1 Functions

The main AEBS function of relevance for this paper is *generating an alert* to warn the flight crew (e.g., via cockpit annunciation) of the proximity of the aircraft to restricted areas of an airport, which are marked by *No Entry* runway

[2] For continuous values, an integration and a probability density function, respectively, replace the summation and the probability mass function. A similar formulation for Eq. (1) is also referenced as *true error* in [10], *probability of misclassification per random input* for classifiers in [11], and is the complement of the *probability of a successful prediction* in [12].

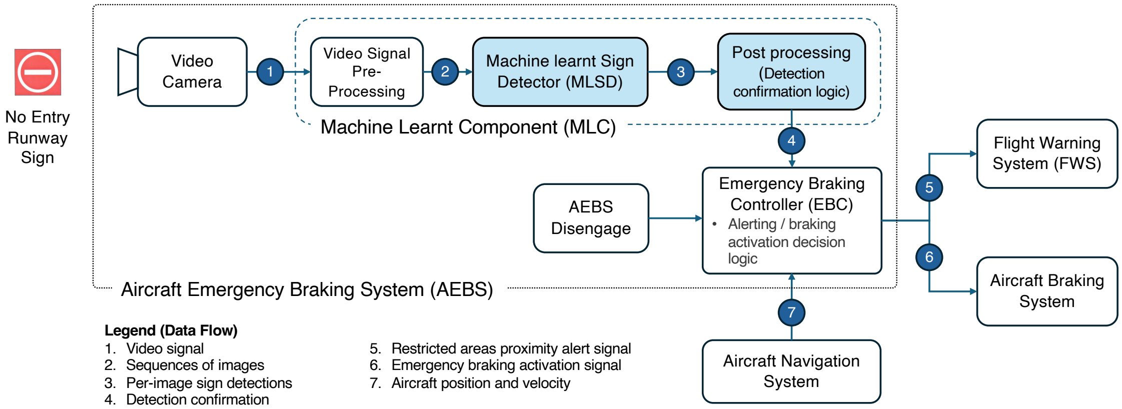
No Entry Runway Sign
Video Camera
Video Signal Pre-Processing
Machine learnt Sign Detector (MLSD)
Post processing (Detection confirmation logic)
Machine Learnt Component (MLC)
AEBS Disengage
Emergency Braking Controller (EBC)
• Alerting / braking activation decision logic
Aircraft Emergency Braking System (AEBS)
Flight Warning System (FWS)
Aircraft Braking System
Aircraft Navigation System
1
2
3
4
5
6
7
Legend (Data Flow)
1. Video signal
2. Sequences of images
3. Per-image sign detections
4. Detection confirmation
5. Restricted areas proximity alert signal
6. Emergency braking activation signal
7. Aircraft position and velocity

Figure 1: Aircraft Emergency Braking System (AEBS) and its machine learnt component (MLC), adapted from [5].

(NER) signs. A sub-function of this alerting function allocated to the MLC is *NER sign detection and classification*. Note that the *emergency braking function* of the AEBS is not in scope for this paper and, as such, affects the safety assessment described later (see Sections 3.2, and 4.2)

3.1.2 Machine Learnt Component

As shown in Fig. 1, The MLC comprises a machine learnt sign detector (MLSD) and its related pre- and post-processing functionality. The MLSD is an implementation of an MLM on target hardware. Here, the MLM is a deep convolutional neural network trained to detect and classify NER signs using supervised, offline learning.

The MLSD inputs (data flow 2) are sequences of images produced after pre-processing the video signal (data flow 1) from an aircraft mounted, forward facing video camera. Video signal pre-processing represents the functionality necessary for the runtime consistency of the types of inputs that the MLSD receives in use, and those on which it is trained offline. The MLSD responses are a sequence of per image detections or non-detections (data flow 3), corresponding to the input image sequence. Those responses undergo post-processing, a key aspect of which is to confirm or reject confirmation of the detection of an NER sign in a *detection vector*, i.e., a fixed size sub-sequence created from the sequence of MLSD responses.

The confirmation of NER sign detection from the post-processing (data flow 4) is then used by the emergency braking controller (EBC) to send a *restricted areas proximity* (RAP) alert (data flow 5) to the flight warning system (FWS), or an emergency braking activation signal (data flow 6) to the aircraft braking system. As previously mentioned, we do not consider the latter for the rest of this paper.

It is worth noting that the post-processing as shown in Fig. 1 is closely coupled to the NER sign detection sub-function. Hence, it is an integral and inseparable element of the MLC. However, in [5] this post-processing is treated as an element of the EBC and referred to as *tracking*, with its failure considered to be the *failure to track NER signs* (also see Section 3.2). Although, it is in fact *detection confirmation*, the term we will use henceforth, rather than true tracking. The detection confirmation logic uses a *confirmation threshold* (i.e., a required number of true per image detections in the detection vector) to confirm that an NER sign has indeed been detected when one exists. This confirmation does not require a specific order of detections in the detection vector.

The detection vector size ($n = 12$) is determined by: (i) the *detection window period* (the time in which the MLSD must detect an NER sign and raise an alert, so that the aircraft can then be safely decelerated and halted either by the pilot or by automation), and (ii) the *detection frequency* (the rate at which the MLSD produces per image detections). Those parameters, in turn, depend on various characteristics of the AEBS, the crew, and the aircraft, which include: (a) the maximum taxiing speed ($30\,\mathrm{kn} \approx 15.43\,\mathrm{ms}^{-1}$), (b) the maximum deceleration ($6\,\mathrm{ms}^{-2}$), (c) the pilot reaction time ($3\,\mathrm{s}$), and (d) the maximum distance from which a detection is required ($85\,\mathrm{m}$). We do not repeat the derivation of those parameters, previously detailed in [5], as it is not required or relevant for this paper. We also note that although the AEBS shown in Fig. 1 modifies and adapts the original from [5], it does not alter those parameter values or their derivation.

3.2 Safety Assessment

The safety effects for which the AEBS is a preventative safety barrier are: (i) an inadvertent incursion into a prohibited area, such as a taxiway meant to be used in a given direction; and (ii) an excursion from an aircraft movement surface onto one not meant for aircraft, such as an intersecting roadway.

As mentioned earlier (Section 3.1), in this paper the scope of the intended use of the AEBS is mainly pilot assistance, even though it includes the capacity for automatic intervention when there is a RAP violation. Thus, the primary safety barrier is still piloting procedures in the runway environment, i.e., the pilot visually acquires NER signs whilst taxiing, and decelerates upon approaching a restricted area. As such, the AEBS serves as an *additional* protection layer, e.g., by providing a RAP alert that will warn the crew if they are distracted. This consideration influences the criticality assigned to the failure conditions of the AEBS function.

An FHA and PSSA for the AEBS have been given previously in [5], which we summarize next, to contextualize the rest of the paper. Specifically, the AEBS functional failure conditions of interest are `LossProxAlrt`: *Loss of RAP alert (crew unaware)*, and `ProxAlertMalfn`: *Malfunction of RAP alert*, each of which are assigned a MINOR severity and a QSO of 10^{-3} pfh, as per the FHA in [5]. Additionally, the PSSA invokes a quantitative FTA [5] to relate `LossProxAlrt` to so-called *ML performance failures*, in particular a *failure of the EBC to track NER signs due to MLC false negatives* allocating to it a QSO of 4×10^{-4} per flight. That target is then halved to account for the assumptions of encountering an average of 2 NER signs per flight, and an average flight duration of 4h, resulting in an effective QSO of 2×10^{-4} per flight for the MLC.

4 Methodology

4.1 Assumptions

To simplify the illustration of the proposed method, we assume the following:

(1) the camera in the AEBS is functional, operating normally, calibrated, stably mounted, and faithfully captures and transmits the environmental scene as a sequence of images;

(2) the environmental scene does not contain other signs or objects that could be mistaken as an NER sign;

(3) there are no transmission errors in the data flow from the video camera through the pre-processing, the MLSD, the post-processing, and the EBC, to the FWS, so that the data transmitted are uncorrupted and have the correct temporal order as captured by the video camera; and

(4) pre-processing does not introduce undesired information into the image stream, e.g., adversarial transformations.

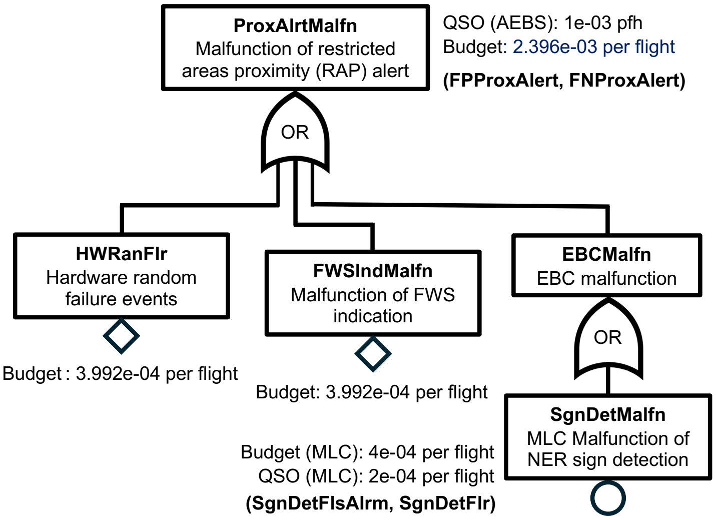
ProxAlrtMalfn
Malfunction of restricted areas proximity (RAP) alert
QSO (AEBS): 1e-03 pfh
Budget: 2.396e-03 per flight
(FPProxAlert, FNProxAlert)
OR
HWRanFlr
Hardware random failure events
Budget : 3.992e-04 per flight
FWSIndMalfn
Malfunction of FWS indication
Budget: 3.992e-04 per flight
EBCMalfn
EBC malfunction
OR
SgnDetMalfn
MLC Malfunction of NER sign detection
Budget (MLC): 4e-04 per flight
QSO (MLC): 2e-04 per flight
(SgnDetFlsAlrm, SgnDetFlr)

Figure 2: Fault tree relating the malfunction of the RAP alert failure condition of the AEBS, to MLC malfunction.

4.2 Revised PSSA and QSO Allocation

The adaptation of the AEBS (see Section 2.2.2, and Fig. 1) from the original architecture in [5] induces modifications to the previously mentioned safety assessment. Additionally we identify some corrections to the FTA in [5]. Fig. 2 shows a revised fault tree for the failure condition `ProxAlertMalfn`, reflecting the following combination of functional failures.

First, `ProxAlertMalfn` can be specialized as two mutually exclusive states: `FPProxAlrt`: *Inadvertent RAP alert (alert issued when not required)*, and `FNProxAlrt`: *Missing RAP alert (alert not issued when required).* In [5], only the former has been identified in the FHA as a failure condition, whereas the latter has been incorrectly considered as equivalent to `LossProxAlrt` in the FTA. Indeed, `FNProxAlrt` can occur when the AEBS and FWS are both operational and available.

Next, from a functional flow standpoint `ProxAlertMalfn` results from a combination of:

(i) `FWSIndMalfn`: *Malfunction of the FWS indication*,
(ii) errors in the FWS alerting logic, or
(iii) `EBCMalfn`: *EBC malfunction*.

`EBCMalfn` can itself result from errors in the alerting activation decision logic in the EBC, or from the input to the EBC (data flow 4 in Fig. 1), reflected as `SgnDetMalfn`: *MLC malfunction of NER sign detection*. That, in turn, manifests as one of two mutually exclusive states[3], i.e., `SgnDetFlsAlrm`: *False confirmation of an NER sign* (a false positive), and `SgnDetFlr`: *Failure to confirm detection of the NER sign* (a false negative).

Per the recommended practice [1], the fault tree in Fig. 2 excludes events corresponding to errors in conventional software, i.e., logic errors in the EBC and the FWS. Additionally, it includes `HWRanFlr`: *Hardware random failure events*, to aggregate and abstract *other* hardware failures that can also lead to the top event. We also include `SgnDetMalfn` in the fault tree, noting that this basic event represents *insufficient MLC performance* rather than a hardware random failure. This is a departure from the conventional practice, justified by the discussion in Section 2.2.5.

For convenience and comparison to the prior literature, we retain the failure probability budgets and QSOs from [5] for both the top event, `ProxAlertMalfn`, and the basic event of the malfunction of the MLC, `SgnDetMalfn`, as shown in Fig. 2. It can be easily confirmed that the probability budgets as shown are correct with respect to the fault tree logic.

Thus, as indicated in Section 3.2, the effective QSO for `SgnDetMalfn` is 2×10^{-4} per taxi operation. Also note that changes to these budgets do not affect the discussion that follows on the proposed method for deriving performance requirements; however the concrete requirements will indeed change.

4.3 Scope of MLC Behavior

Again, for convenience, and ready comparison to [5], in what follows, we mainly focus on the taxiing scenarios where an NER sign is actually present. As such, the failure condition `ProxAlertMalfn` effectively presents as the state `FNProxAlrt` (i.e., RAP alert not issued when required), and, likewise, the basic event `SgnDetMalfn` is the state `SgnDetFlr` (i.e., a failure of the MLC to confirm detection of the NER sign). Together with the earlier assumptions (Section 4.1), the scope of MLC behavior and the subsequent analysis for developing safety-related performance requirements for this paper is constrained as follows:

- When the operating environment contains an NER sign, then the responses of the MLSD (see Fig. 1) to an input image containing that NER sign are either: (i) a *hit*, i.e., a correct (true positive) detection of the NER sign (including correct bounding boxes and class labels), or (ii) a *miss*, i.e., all MLSD responses that are not a hit. Effectively, a miss is only a false negative, since false positives or false classifications cannot be produced in scenarios where an NER sign is actually present in the environment.
- Depending on the number of hits and misses determining the confirmation threshold in the detection vector, the detection confirmation logic either confirms an NER sign detection, or it does not confirm an NER sign detection.
- Thus, in all taxiing scenarios where an NER sign is present in the operating environment, when the post-processing does not confirm a sign detection, it represents the occurrence of an MLC malfunction in the state `SgnDetFlr`, with an effective QSO of 2×10^{-4} per flight (taxi operation).

[3] In general, sign detection malfunctions are *false positives* or *false classifications* where, for example, either one type of runway sign is misclassified as a different type of sign, or as not a sign (i.e., a *false negative*). However, in this example, since the MLM is a binary classifier trained specifically for NER sign detection, the MLC produces a Boolean confirmation response.

4.4 From Safety Objectives to Safety-related Performance

The QSO allocated to `SgnDetMalfn` is the starting point for deriving the MLC performance requirements and metrics in the AEBS. As clarified above, that event is based on the per image detections received from the MLSD, in the detection vector, during post-processing.

Specifically, according to the detection confirmation logic, a non-detection occurs when the detection vector contains fewer per image hits than the minimum permissible number of hits required to confirm detection. In other words, when an NER sign is present, to avoid `SgnDetFlr`:

(i) the detection vector must contain at least as many hits as the confirmation threshold; and
(ii) the confirmation threshold should be defined such that the probability of not confirming a detection must be lower than the QSO allocated to `SgnDetFlr`.

Note that a related concept of *rejection threshold* can be considered that results in not confirming that an NER sign has been detected. Thus, the confirmation (or rejection) threshold is a parameter relevant for safety-related performance.

Additionally, when an NER sign is present in the operating environment and the detection vector contains more per image misses than hits, it suggests that the MLSD has a larger than required *per image probability of non-detection* (equivalently, the *per image miss probability*) leading to the rejection threshold being satisfied. Thus, the per image miss probability is a safety-related model performance metric, and to avoid `SgnDetFlr` it should be defined such that the rejection threshold is not met (or, equivalently, the confirmation threshold is met).

5 Safety-related Performance Requirements

We now formalize the preceding intuition as an abstraction of the required behavior (Fig. 3), from which we formulate safety-related performance metrics and requirements for the MLC and its underlying MLM. The focus is on specifying requirements rather than verifying that the requirements have been met.

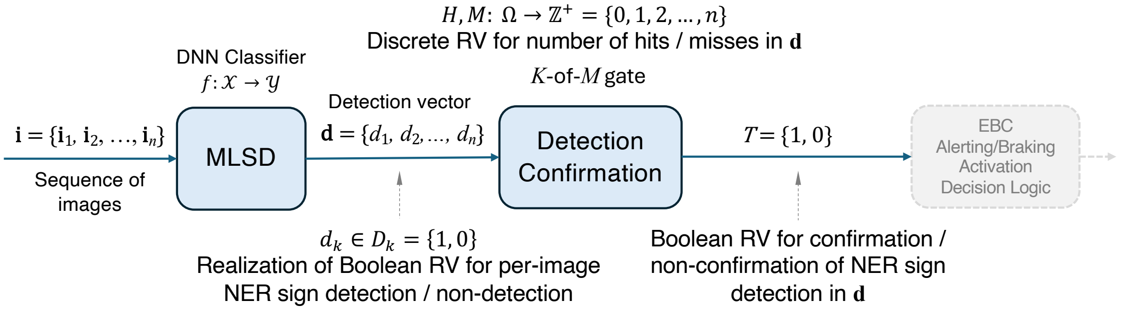
$H, M\colon \Omega \rightarrow \mathbb{Z}^+ = \{0, 1, 2, \ldots, n\}$
Discrete RV for number of hits / misses in **d**
DNN Classifier
$f\colon \mathcal{X} \rightarrow \mathcal{Y}$
K-of-M gate
Detection vector
$\mathbf{i} = \{\mathbf{i}_1, \mathbf{i}_2, \ldots, \mathbf{i}_n\}$
MLSD
$\mathbf{d} = \{d_1, d_2, \ldots, d_n\}$
Detection Confirmation
$T = \{1, 0\}$
EBC Alerting/Braking Activation Decision Logic
Sequence of images
$d_k \in D_k = \{1, 0\}$
Realization of Boolean RV for per-image NER sign detection / non-detection
Boolean RV for confirmation / non-confirmation of NER sign detection in **d**

Figure 3: Abstraction to describe the required behavior for NER sign detection using the MLC.

5.1 Abstraction of Required Behavior

Let $T = \{0, 1\}$ be a Boolean random variable (RV) for the event of an MLC response, the output of the post-processing detection confirmation logic. Those responses are either a confirmation of detection of an NER sign, i.e., the event $(T = 1)$, or the malfunction `SgnDetMalfn`, i.e., the event $(T = 0)$. As clarified in Sections 4.2 and 4.3, `SgnDetMalfn` occurs as the state `SgnDetFlr`, i.e., a failure to confirm detection of the NER sign. Hence,

$$\texttt{SgnDetFlr} \overset{\text{def}}{=} (T = 0)$$

Let the QSO allocated to `SgnDetMalfn` be q_{tr}. Thus, a concrete safety-related MLC performance requirement for NER sign detection, based on the allocation from the PSSA process (specifically, the FTA in Fig. 2), is:

Requirement 1 (MLC Safety Performance)**.** *The average probability of non-detection of an NER sign per taxi operation shall be less than* q_{tr}*, i.e.,* $\Pr(T = 0) < q_{\text{tr}} \leftarrow 2 \times 10^{-4}$

We can specify an analogous requirement on the MLC *functional performance* as:

Requirement 2 (MLC Functional Performance)**.** *The probability of detecting an NER sign shall be at least* $(1 - q_{\text{tr}})$*, i.e.,* $\Pr(T = 1) \geq (1 - q_{\text{tr}}) \leftarrow 0.9998$

From Fig. 3, the detection vector, $\mathbf{d} = \{d_1, d_2, \ldots, d_n\}$, of size n, is a finite sequence of responses produced by the MLSD, f, to a sequence of input images $\{\mathbf{i}_j\}_{j=1}^{n}$. Here, $d_j \in \{1, 0\}$ is the realization of D_j, a Boolean RV representing the event of the j^{th} response of f to the j^{th} input image $\mathbf{i}_j$. If $(D_j = 1)$, $(D_j = 0)$ represent a hit and a miss, respectively, then whenever there is a hit in $\mathbf{d}$, $d_j = 1$, otherwise $d_j = 0$.

Let the confirmation and rejection thresholds be $x_{\min}$ and $y_{\min}$, respectively. Also let H, M be the discrete RVs for the number of hits and misses, respectively, whose realizations are $h, m \in \{0, 1, 2, \ldots, n\}$. As clarified earlier (Section 2.2.2 and 4.4), the post-processing confirms that an NER sign has been detected when $h \geq x_{\min}$. Moreover, since hits do not need to occur in a specific order in $\mathbf{d}$ for a detection confirmation, the corresponding logic is a K-of-M *gate*, where $K = x_{\min}$ and $M = n$.

We can now readily confirm that h is the sum of the individual detections in $\mathbf{d}$, and formalize the detection confirmation logic as: $\forall \mathbf{d}, (h = \sum_{j=1}^{n} d_j) \geq x_{\min} \Rightarrow (T = 1)$. We will concretize this as a requirement next, in Section 5.2.

Since $\mathbf{d}$ contains a combination of hits and misses, we have $n = h + m$, and when $h = x_{\min}$, then $m = (n - x_{\min})$ represents the maximum permissible per image misses in $\mathbf{d}$ that still results in a detection confirmation. Hence, an additional miss will result in a failure to confirm detection, so that $y_{\min} = (n - x_{\min}) + 1$.

Now, assume that a hit or miss response of f is the result of a Bernoulli trial, and that each $D_j \in \mathbf{d}$ is independent and identically distributed (IID)[4]. Then, let the *per image hit probability*, $\Pr(D_j = 1) = p_{\text{hit}}$, so that the *per image miss probability*, $\Pr(D_j = 0) = p_{\text{miss}} = 1 - p_{\text{hit}}$.

We have that $\mathbf{d}$ is a realization of a *Bernoulli process*, i.e., a sequence formed by the result of n Bernoulli trials in which there are h events such that $(D_j = 1)$ and m events such that $(D_j = 0)$. Since the sum of the RVs of a Bernoulli process is another RV that follows a binomial distribution, $H \sim \texttt{Binomial}(n, p_{\text{hit}})$, and the probability of at least h hits is

$$\Pr(H \geq h) = \sum_{i=h}^{n} \binom{n}{i} p_{\text{hit}}^{i} (1 - p_{\text{hit}})^{n-i} \tag{2}$$

Hence, the probability of confirming an NER sign detection is

$$\Pr(T = 1) = \Pr(H \geq x_{\min}) = \sum_{i=x_{\min}}^{n} \binom{n}{i} p_{\text{hit}}^{i} (1 - p_{\text{hit}})^{n-1} \tag{3}$$

Then, the probability of failure to confirm detection of the NER sign, $\Pr(T = 0)$, is $1 - \Pr(T = 1)$, which we formulate in terms of M, n, $y_{\min}$, and p_{miss}. That is,

$$\Pr(T = 0) = \Pr(M \geq y_{\min}) = \sum_{i=y_{\min}}^{n} \binom{n}{i} p_{\text{miss}}^{i} (1 - p_{\text{miss}})^{n-1} \tag{4}$$

5.2 Concrete Performance Requirements

To establish concrete requirements for p_{hit}, p_{miss}, $x_{\min}$ and $y_{\min}$, we solve either of (3) and (4) such that reqs. 1 or 2, respectively, are satisfied. Fig. 4 shows a graphical solution, varying $\Pr(T = 0)$ on a logarithmic scale, for the rejection thresholds $12 \geq y_{\min} > m \in [4, 11]$, and a range of $p_{\text{miss}} = [0, 0.5]$.

The dotted horizontal line in Fig. 4 is the QSO for failing to confirm NER sign detection. As shown, the QSO is not met in region A, but is satisfied in region in B for rejection thresholds $y_{\min} = 12 \geq m > 5$. The region C, between the two vertical dotted lines, is a sub-region of B, giving a candidate range for $p_{\text{miss}} \approx [0.087, 0.177]$ and $y_{\min} = [6, 8]$, respectively. Then, together with the previous discussion (Section 5.1), we obtain a range for $p_{\text{hit}} \approx [0.823, 0.913]$ and $x_{\min} = [5, 7]$.

From Fig. 4, we have $p'_{\text{miss}} \approx 0.124$, where the QSO is exactly q_{tr} for $x_{\min} = 6$. We can now select, say, $y_{\min} > 6$, and $p_{\text{miss}} = 0.1$, so that $x_{\min} = 6$ and $p_{\text{hit}} = 0.9$. Then we can specify additional concrete performance requirements for the MLC and its elements, namely the MLSD, and its post-processing.

First, the concrete requirement based on the formalization of the detection confirmation logic is:

[4] Section 6.2 justifies these assumptions and discusses their implications.

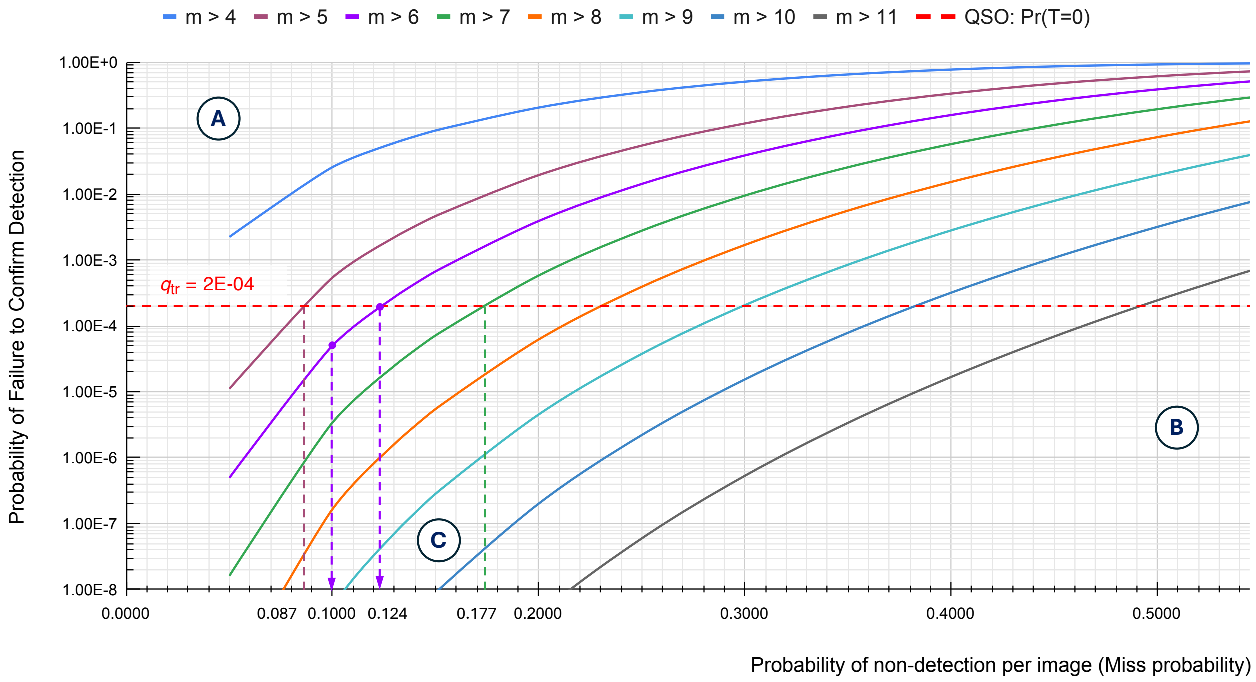

Figure 4: Varying $\Pr(T = 0)$ on the y-axis, with p_{miss} on the x-axis, for different values of m, determined from (3).

Requirement 3 (MLC Detection Confirmation)**.** *The MLC post-processing shall confirm an NER sign detection whenever there are at least* 6 *detections in any detection vector, i.e.,* $\forall \mathbf{d}, x_{\min} \geq 6 \Rightarrow (T = 1)$

We may equivalently specify the dual of Req. 3 specifying the rejection of sign detection confirmation based on the rejection threshold as:

Requirement 4 (MLC Reject Detection Confirmation)**.** *The MLC post-processing shall reject confirmation of an NER sign detection whenever there are at least* 7 *non-detections in any detection vector, i.e.,* $\forall \mathbf{d}, y_{\min} \geq 7 \Rightarrow (T = 0)$

Then, similar to Req. 1, the required MLSD safety performance in terms of the respective miss probability is:

Requirement 5 (MLSD Safety Performance)**.** *The MLSD shall have a per image probability of non-detection of an NER sign of at most* 0.1*, i.e.,* $\forall d_j \in \mathbf{d}; \Pr(D_j = 0) = p_{\text{miss}} \leq 0.1$

As earlier, we can give an analogous requirement for MLSD functional performance in terms of p_{hit} as:

Requirement 6 (MLSD Functional Performance)**.** *The MLSD shall have a per image probability of detection of an NER sign of at least* 0.9*, i.e.,* $\forall d_j \in \mathbf{d}; \Pr(D_j = 1) = p_{\text{hit}} \geq 0.9$

We additionally specify MLSD safety performance in terms of a *tolerable miss ratio* metric, m_{t}, i.e., the allowable proportion of missed detections per detection vector, which we compute as: $m_{\text{t}} = (\mu_m + \sigma_m)/n \approx 0.187$, where $\mu_m = np_{\text{miss}}$ is the mean, and $\sigma_m^2 = np_{\text{miss}}(1 - p_{\text{miss}})$ is the variance, respectively, of $M \sim \texttt{Binomial}(n, p_{\text{miss}})$. Hence:

Requirement 7 (MLC Safety Performance – Miss Ratio)**.** *The tolerable miss ratio for the MLSD shall not exceed* 0.187*, i.e.,* $m_{\text{t}} \leq 0.187$

5.3 Generalization Performance Requirements

Recalling the discussion in Section 2.2.5, when the MLM failure probability exceeds its requirement as derived from the safety objectives, its generalization performance is insufficient. In other words, the minimum required generalization performance requirement is related to the maximum tolerable MLM failure probability, which we now characterize in terms of the model *generalization error* and *generalization gap*.

5.3.1 Generalization Error

The generalization error $R_p(f)$ for an MLM f, also known as the *population risk*, is defined as the expected value of a suitable *loss function*, $\boldsymbol{\ell}(f(\mathbf{x}), \mathbf{y})$, evaluated over the joint distribution $\mathrm{Pr}_{X,Y}(x, y)$ of the input and output spaces for f. Thus,

$$R_p(f) \overset{\text{def}}{=} \mathbb{E}_{(x,y)\sim\mathrm{Pr}_{X,Y}} \boldsymbol{\ell}(f(\mathbf{x}), \mathbf{y}) \tag{5}$$

For binary classification, a commonly used loss function is the so-called *zero-one* loss, defined as

$$\boldsymbol{\ell}(f(\mathbf{x}, \mathbf{y})) \overset{\text{def}}{=} \mathbf{1}_f(x) = \begin{cases} 1 & \text{when } f(\mathbf{x}) \neq \mathbf{y} \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

Theorem 1. *The generalization error for an MLM performing binary classification is exactly its failure probability under the zero-one loss.*

Proof. We have

$$\begin{aligned}
R_p(f) &= \mathbb{E}_{(x,y)\sim\mathrm{Pr}_{X,Y}} \mathbf{1}_f(x) && \ldots\text{By substituting (6) into (5)} && \text{(6a)}\\
&= \sum_{x,y} \mathbf{1}_f(x)\mathrm{Pr}_{X,Y}(x, y) && \ldots\text{From the definition of expectation} && \text{(6b)}\\
&= \sum_{x} \mathbf{1}_f(x) \sum_{y} \mathrm{Pr}_{X,Y}(x, y) && \ldots\text{Distributive property} && \text{(6c)}\\
&= \sum_{x} \mathbf{1}_f(x)\mathrm{Pr}_{X}(x) && \ldots\text{By marginalization over y} && \text{(6d)}\\
&= \mathrm{Pr}(f(\mathbf{x}) \neq \mathbf{y}) && \ldots\text{From (1)} && \text{(7)}
\end{aligned}$$

□

Now, if $\mathbf{x}$ is an input image containing an NER sign, $\mathbf{i}_j$, then as clarified in Section 5.1, the required MLSD response $\mathbf{y}$ is $d_j = 1$ in $\mathbf{d}$, i.e., $(D_j = 1)$; hence, (7) is equivalent to the per image probability of non-detection of an NER sign, $\mathrm{Pr}(D_j = 0)$. From Fig. 4, $\mathrm{Pr}(D_j = 0)$ attains its maximum tolerable value when q_{tr} is met; therefore $R_p(f) = p'_{\mathrm{miss}}$. Thus,

Requirement 8 (MLSD Generalization Performance)**.** *The MLSD shall have a generalization error of at most* 0.124*, i.e.,* $R_p(f) \leq p'_{\mathrm{miss}} \leftarrow 0.124$

However, neither the joint nor the input distribution may be exactly known. Thus, although we can require R_p to be p'_{miss}, its *true* value cannot be determined. Instead, in practice, $R_p(f)$ is estimated using the *empirical test risk* metric, $R_{\mathrm{e}}(f, \mathcal{D}_{\mathrm{test}})$, under the requirement that the test data $\mathcal{D}_{\mathrm{test}}$ (as well as the training data) used to learn f are sampled from a *representative* joint distribution. For the zero-one loss the empirical risk measured on dataset $\mathcal{D}$ is, in fact, the *false classification rate* performance metric [8]. Recalling Section 4.3, the false classification rate for f is the *false negative rate*, $\mathtt{FNR}(f, \mathcal{D})$. Thus, we can refine Reqs. 5 and 8 as:

Requirement 9 (MLSD Performance – False Negative Rate in Test)**.** *The MLSD shall have a false negative rate in test of at most* 0.1*, i.e.,* $\mathtt{FNR}(f, \mathcal{D}_{\mathrm{test}}) \leq p_{\mathrm{miss}} \leftarrow 0.1$

Additionally, the *true positive rate* performance metric (also known as *sensitivity* or *recall*), $\mathtt{TPR}(f, \mathcal{D})$, measured on a dataset $\mathcal{D}$, is the dual of the false negative rate. Thus,

Requirement 10 (MLSD Performance – Recall in Test)**.** *The MLSD shall have a recall in test of at least* 0.9*, i.e.,* $\mathtt{TPR}(f, \mathcal{D}_{\mathrm{test}}) \geq (1 - p_{\mathrm{miss}}) \leftarrow 0.9$

5.3.2 Generalization Gap

The *empirical training risk*, $R_{\mathrm{e}}(f, \mathcal{D}_{\mathrm{train}})$, is an analogous metric to the empirical test risk. The difference between the two gives an estimate of the *generalization gap*, which is, itself, the difference between the generalization error and the empirical training risk, i.e.,

$$R_p(f) - R_{\mathrm{e}}(f, \mathcal{D}_{\mathrm{train}}) \approx R_{\mathrm{e}}(f, \mathcal{D}_{\mathrm{test}}) - R_{\mathrm{e}}(f, \mathcal{D}_{\mathrm{train}}) \tag{8}$$

We can now give a probabilistic upper bound δ to the generalization gap (or to its estimate) using *Hoeffding's inequality* and the *union bound* theorems [8]. Thus, for data $\mathcal{D}$ comprising η samples and a tolerance ϵ in the generalization gap, we have:

$$\Pr(|R_p(f) - R_{\mathrm{e}}(f, \mathcal{D})| > \epsilon) \leq \delta = 2e^{-2\eta\epsilon^2} \tag{9}$$

which can also be rearranged as:

$$\eta \geq \frac{1}{2\epsilon^2} \ln\left(\frac{2}{\delta}\right) \tag{10}$$

Note that some of the available literature refers to ϵ as *accuracy*, and δ as *confidence*. To avoid a misinterpretation of those terms as used in the contexts of aircraft certification, and system safety, versus ML, we refer to ϵ as the *tolerance* and to δ as the probabilistic upper bound instead.

The minimum number of independent samples required to satisfy the bound can be determined from (10), by selecting the desired tolerance and the probabilistic upper bound. For example, select: (i) $\delta = 1 \times 10^{-3}$, proportional to the order of magnitude of the QSO, and (ii) $\epsilon = \mathtt{S_M}(p'_{\mathrm{miss}} - p_{\mathrm{miss}})$, where $\mathtt{S_M}$ is a margin of safety. The reasoning here is that a tolerance greater than the difference in the required generalization error and the required false negative rate, $(p'_{\mathrm{miss}} - p_{\mathrm{miss}})$, results in a failed detection confirmation. Thus, selecting $\mathtt{S_M} = 0.5$, and from Fig. 4, $p'_{\mathrm{miss}} \approx 0.124 \Rightarrow \epsilon = 0.012$, therefore $\eta \geq 26393$ independent samples (drawn from a representative distribution).

Depending on whether this procedure is applied to the generalization gap or to its estimate, we can upper bound either of the two and derive the sample sizes of the training and test datasets required at the chosen tolerance. Thus, additional testing-related requirements can then be specified (not given here).

6 Discussion

6.1 Rationale for Assurance of Validity

A robust validation of the proposed method and the consequent performance requirements (Sections 4 and 5) requires a careful research design, which is out of scope for this paper, and an avenue for future work. Instead, this section provides rationale to justify why the proposed method and the resulting requirements are a valid step to relate system-level QSOs and the performance requirements of machine learnt functionality.

6.1.1 Suitability of the System Architecture and QSOs

The MLC and its organization in the AEBS (Fig. 1) represent a *single channel architecture*. That is, a loss or malfunction of any element of the channel compromises the entire channel. Hence it is the weakest from the standpoint of both reliability and safety. When decomposing and allocating the QSO to be achieved by such an architecture to its elements (including an MLC), the allocated QSOs are more conservative than they would be for alternative architectures, e.g., with redundancy, or diversity. In that sense, given the intended use and the safety assessment (Section 3), the chosen architecture and the QSO for the MLC are the tolerable worst-case. Therefore they are appropriate and sufficient as a starting point to formulate a conservative set of MLC performance requirements.

6.1.2 Suitability of the Performance Requirements

We model the probability of failure of the MLSD as the limiting relative frequency of incorrect responses to random image inputs from the input space, as given by Eqs. (1), (5) – (7). As such, MLSD failure behavior, as modeled, is equivalent to random failure.

Then, in the FTA for the AEBS (Fig. 2), we capture MLC malfunction as the EBC malfunction basic event, computing its failure probability as in Section 5.1. This is analogous to the result of a quantitative FTA for a K-of-M gate (also known as a *voting gate*), whose basic events are each of the per image responses of the MLSD in the detection vector. Here, a per image non-detection, i.e., an incorrect response, is equivalent to the random failure of the corresponding basic event with a constant failure probability p_{miss}. Thus, the binomial model for detection confirmation (Section 5.1 and Fig. 3) abstracts MLC malfunction also as a random failure.

Now, as clarified in Sections 2.2.3 and 2.2.4, the MLSD is both deterministic and systematic in its behavior. Furthermore, the MLSD implements a deep convolutional neural network (Section 3.1.2), which has a *feedforward* neural architecture, i.e., there are no feedback loops between its neurons in the network layers. Thus, it is also *stateless*, with the responses depending only on the current inputs, and not on the history of inputs or prior responses. Furthermore, detection confirmation is a deterministic rule-based decision. Together, it implies that, under the stated assumptions and scope (Sections 4.1 and 4.3), for any input, and input sequence subsequently formed: (i) the ideal (best case) MLC behavior is a systematically correct response due to perfect generalization of the MLSD and a deterministic choice of NER sign detection confirmation; and (ii) the worst case MLC behavior is a systematically incorrect response due to consistent MLSD failure followed by a deterministic choice rejecting NER sign detection confirmation.

6.1.3 Validity

Since random behavior lies between the worst case and ideal behavior, and since the concrete performance requirements defined based on random behavior (Section 5.2) have been mathematically derived from the allocated QSO, we can conclude that: (i) the performance requirements as specified meet the allocated QSO by construction; (ii) any systematic MLC behavior up to the ideal, verified to meet or exceed the specified requirements will also meet the allocated QSO; and (iii) the requirements as defined are the minimum required, being the tolerable worst-case.

6.2 Threats to Validity

First, correct application of the FTA for QSO allocation (Fig. 2) may potentially challenge the rationale for the suitability of the allocated QSO (Section 6.1.1). Specifically, the recommended practice for FTA [1] only considers hardware failure basic events, with associated failure rates, rather than basic events representing insufficient MLC performance, e.g., EBC malfunction with a failure probability.

However, quantitative FTA admits computation with failure probabilities, and Section 2.2.5 clarifies why a probability of failure can indeed be assigned to the MLC malfunction basic event, thus justifying its inclusion in the FTA. Additionally, note that the purpose of the FTA as in Section 4.2 is to *specify* requirements rather than to *verify* that they have been met. As such, we contend that the way we have applied FTA is sound. Furthermore, although changes to the specific value of the system level QSO can change both the allocated QSO and the concrete performance requirements, the proposed method to develop those requirements, itself, is unaffected.

Next, the constraints for applying a binomial model may potentially challenge its use and the associated rationale for the suitability of the resulting performance requirements (Section 6.1.2). We enumerate and substantiate each constraint:

(i) *Fixed number of Bernoulli trials*: Met due to a fixed size detection vector ($n = 12$), and by definition (Section 5.1), with Boolean responses for both the MLSD and the MLC.

(ii) *IID trials*: The input to the MLC, and subsequently to the MLSD, is a temporally ordered sequence of images of the runway scene as captured and transmitted by the video camera. Hence, they are a correlated time series from the same generating process, due to which they are *identically distributed* but *not independent*. The MLSD is systematic, deterministic, and stateless; hence its responses are also identically distributed but not independent. Since the MLSD responses are the inputs to the binomial model, the IID constraint is *not met*.

However, this constraint effectively implies that the trials should be random. Thus, our counter argument here is that maintaining the assumption of IID trials does not invalidate the requirements because: (a) despite abstracting the MLC failure behavior as random, the concrete performance requirements meet the QSO by construction; and (b) as before, we use the abstraction to define the requirements rather than to verify that they have been met, which is when the IID constraint would apply. That is, the MLC must be verified with non-IID data against Requirements 1–10, even though those requirements have been defined assuming IID inputs for post-processing.

(iii) *Constant probability of trial outcomes*: Eqs. (1), (5) – (7) clarify the relationship of the MLSD failure probability to the distribution of its inputs, showing that the former is deterministically related to the latter by the identity function. Since the inputs to the MLSD have been established to be identically distributed, their moments (e.g., their expected values) are also identical and therefore constant (but unknown). Hence the MLSD failure probability is constant.

A concern here is that p_{miss} may change over the long run, due to a drift in the input space distribution. Mitigating the effects of such distribution drift requires carefully describing the ODD (see Section 6.3.1), and consideration of the exposure duration over which the QSO and failure probabilities are expressed, i.e., the duration for which the input distribution is expected to be stable, and where p_{miss} will then be constant.

6.3 Additional Considerations

6.3.1 Relevance of the Operational Design Domain

Defining $\Pr_{X,Y}(x, y)$, the joint distribution of the input and output space, underpins both the ML process and the development of MLC and MLM performance requirements. That induces specific additional considerations on sufficiently characterizing: (i) the *marginal* input space distribution, $\Pr_X(X)$, reflecting the intended operating environment; and (ii) the *conditional* input space distribution, $\Pr_{X|Y}(X|Y)$, which reflects functional intent. Both considerations require, in part, a well-defined and validated ODD from which data must be sampled to meet various data properties [4, 7, 9]. The latter consideration in particular levies requirements on the pre-processing element, or more generally on the system architecture, to assure that the MLC only receives inputs consistent with its defined input space (known as *in-ODD* or *in-domain*) and functional intent (i.e., *in-distribution*), as considered during the ML process.

6.3.2 Robustness Performance

The ML literature treats model robustness separately from generalization performance. However, from a safety standpoint, we contend that MLM failures in general, especially those that result from model fragility under input perturbation or abnormality, stem from an inadequate definition of both the ODD and the corresponding input space distribution. This viewpoint is consistent with how, for example, a lack of robustness in conventional airborne software is treated as a requirements inadequacy [2]. As such, not only normal range inputs but also aberrant and limiting inputs should be considered in the ODD and the corresponding input space distribution when specifying and evaluating MLM failure probability and the associated performance requirements.

Thus, in this paper, although MLSD robustness performance has not been considered, we indicate a potential way forward for future work: as clarified in Section 5.3, the MLSD generalization error $R_p(f)$ equals its maximum tolerable failure probability p'_{miss}. We propose to treat it as a metric of *robust generalization* that considers failures due to both a lack of model robustness and inadequate generalization for previously unseen inputs. Hence, we can have:

$$R_p(f) = R_p^{(r)}(f) + R_p^{(g)}(f)$$

where $R_p^{(r)}(f)$ is the portion of the generalization error apportioned to robustness related failures, and $R_p^{(g)}(f)$ is the remainder. We believe this has the advantage of being able to reuse empirical risk and other related metrics, as in Section 5.3, but for robustness. Future work will thus explore expressing $R_p^{(r)}(f)$ appropriately (e.g., in terms of the relative frequency of abnormal inputs that lead to failures), as well as the relationship to the prevailing robustness related metrics.

6.3.3 Implications for Verification

As previously indicated (Section 6.2), a consequence of using the method described in Section 5 is that the underlying abstraction cannot also be used to verify that the implementation meets the defined requirements. In particular, the assumptions of the binomial model cannot be used to define verification requirements because the IID constraint will not be met.

A second implication relates to the dataset sample size η necessary to meet the bound on the generalization gap and the related tolerance (Section 5.3). Specifically, Eq. (9) applies to any RV that can be bounded and does not depend on the underlying data distribution. Hence, η is a pessimistic worst-case lower bound, which increases quadratically with a smaller tolerance ϵ in the generalization gap. Thus, alternative methods (such as using the Normal distribution approximation to the binomial) could be used to give more favorable sample sizes, subject to the validity of the assumptions of those alternative methods.

7 Concluding Remarks

7.1 Related Work

Our adaptation of the AEBS (Fig. 1) is from [5], which is the closest counterpart to this paper. In Section 3 and 4.2, we clarified the modifications our paper makes to the AEBS and its safety assessment, correcting what we believe is an erroneous application of the FTA in [5]. Additionally, in [5], the Poisson distribution is used, with limited justification, to compute the per image miss probability as $p_{\text{miss}} \leq 0.19$, with the confirmation threshold selected as $x_{\min} = 5$. However, the justification for those choices is weak, although it is acknowledged that a binomial distribution is more precise. In contrast, we give a detailed description of the binomial model for the intended MLC and MLM behaviors, with a range of admissible values for the corresponding parameters (Section 5), also supplying substantiating rationale (Section 6). Additionally, our paper relates the QSO to both MLM generalization and sample size estimates for the test data, thus going further than [5].

The automotive systems domain has progressively considered the relation between system safety and MLM performance. For instance, in the context of detecting pedestrians, reasoning about how evidence of MLC performance contributes to safety assurance relies on showing that a required level of safety-related MLC performance has been attained in a defined operating environment [14]. That, in turn, has been formalized within a safety assurance case framework as an assume-guarantee contract, which invokes quantitative performance requirements that an MLC must meet, under assumptions fulfilled by its inputs. In [15], the evaluation that such safety contracts are satisfied is further explored using *subjective logic*, to account for the uncertainty in the assessment. That work also proposes that uncertainty-aware evaluation of ML performance requirements may provide the mechanisms to define suitable target values for the related metrics, such that they satisfy the system safety objectives, but stops short of clarifying what those mechanisms are.

In the same application context of pedestrian detection, [16] emphasizes the elicitation and analysis of ML safety requirements, and their impact on assurance activities during ML development. That work, in turn, leverages a structured process to determine so-called *validation targets* [17] necessary for assurance of safety of the intended functionality. Validation targets represent the evidence necessary to confirm, amongst other things, safety-related performance of machine learnt models, and the underlying insufficiencies. Both [16] and [17] give examples of machine learnt model performance and robustness requirements that impact system safety, along with rationale that clarifies the choice of specific metrics and their values. However, this clarification is limited as regards the procedures used to determine those metrics, their values, and how they follow from safety objectives.

Assurance cases for safety functions that use supervised ML have recognized that it is a key goal to associate the safety-related properties, metrics and performance of MLCs to the higher level system safety requirements [18]. That goal is supported by lower-level claims of completeness, consistency, and sufficiency of the referenced properties, metrics, and the associated performance limitations. The evidence for each of those, in turn, includes the results of safety analyses, systematic reviewing, safety verification, and—crucially—the definition of (valid) safety-related metrics and performance limitations. Again, indications on how latter is accomplished are notably absent.

Other contemporary research has investigated the derivation of reliability requirements for MLCs. For example, safety-related visual transformations and changes for which human vision performance is unaffected have been used to determine and verify reliability requirements for MLCs used for machine vision [19]. Drawing on the concepts of software *operational profiles* and *probability of failure on demand* (PFD), [11] defines reliability and robustness metrics for DNN classifiers. In [12], conformal prediction is leveraged to give a procedure to derive lower bounds on DNN reliability. Although, in both [11] and [12], reliability modeling is a *bottom-up*, component-level process, relying upon data, and iterative assessment of trained models. The work in this paper is rather a *top-down* method.

A survey of contemporary approaches to specifying safety requirements for MLCs identifies various related considerations to be addressed [20]. However, it does not specifically address how MLC safety requirements, or safety-related MLC performance requirements are to be related to or derived from system safety objectives.

Other aviation guidelines [1–3, 13] do not give examples translating QSOs into item level performance requirements, and are inapplicable for items including ML. There exist minimum operational performance standards (MOPS), and minimum aviation system performance standards (MASPS), that give function and application specific safety-related performance requirements, though they do not consider ML. Thus, our paper aims to mirror such efforts for machine learnt functionality, extending the state of the practice.

Elements of the work in this paper have previously informed the ongoing effort of industry consensus-based standards committees [21] (e.g., EUROCAE WG-114 and SAE G-34), whose members include civil aviation regulators.

However, their work is still in progress and as yet unpublished, hence we are unable to provide more details and clarification contrasting it with this paper.

7.2 Summary and Future Work

The main contribution of this paper is an initial method, with rationale for validity, to mathematically translate QSOs allocated from a system safety assessment into the safety-related performance requirements and the associated metrics for an MLC and its underlying MLM. Using an example of an aircraft emergency braking system that uses a deep neural network for runway sign detection, we have illustrated the method at the system, component, and model levels, showing the relationship to machine learnt model generalization, and sample size for test data. To the best of our knowledge, this paper is the first to systematically relate system safety and safety-related machine learnt model performance requirements.

There are several avenues to further improve upon this initial work: first, we intend to extend our method to address robustness performance (Section 6.3), for example, by relaxing the assumptions on pre-processing (Section 4.1). Next, we also aim to address per image false positives and false classifications, without which the current requirements are more optimistic than they should be. A candidate approach here, is to use a multinomial model, whilst also exploring Bayesian approaches, e.g., with beta and Dirichlet priors, to capture and specify the uncertainty in the performance metrics. Additionally, the following are key to a broader applicability of our method: (i) addressing applications involving regression and multi-class classification problems, also considering other types of loss functions, generalization bounds, and metrics; and (ii) addressing alternative procedures for detection confirmation, e.g., using longer and/or multiple detection sequences.

Acknowledgment

We thank various members of EUROCAE WG-114 and SAE G-34 for numerous fruitful discussions on the topic of this paper.

References

[1] S-18, Aircraft And System Development And Safety Assessment Committee, “Guidelines and Methods for Conducting the Safety Assessment Process on Civil Aircraft, Systems, and Equipment,” Aerospace Recommended Practice ARP4761 Rev. A, SAE International, Dec. 2023.

[2] RTCA SC-205 and EUROCAE WG-71, “Software Considerations in Airborne Systems and Equipment Certification,” DO-178C / ED-12C, Dec. 2011.

[3] RTCA SC-180 and EUROCAE WG-46, “Design Assurance Guidance for Airborne Electronic Hardware,” DO-254 / ED-80, April 2000.

[4] EASA, “EASA Artificial Intelligence (AI) Concept Paper Issue 2: Guidance for Level 1 and 2 Machine Learning Applications,” March 2024. [Online]. Available: https://www.easa.europa.eu/en/downloads/139504/en

[5] K. Dmitriev, J. Rhein, L. Beller, J. Bröcker, E. Huber, J. Schumann, and F. Holzapfel, “Safety Assessment of a Machine Learning-Based Aircraft Emergency Braking System: A Case Study,” in *2024 AIAA DATC/IEEE 43rd Digital Avionics Systems Conference (DASC)*, 2024. doi:10.1109/DASC62030.2024.10749696

[6] Federal Aviation Administration, Small Airplane Directorate, ACE-100, “System Safety Analysis and Assessment for Part 23 Airplanes,” Advisory Circular AC 25.1309-1E, Federal Aviation Administration, November 2011. [Online]. Available: https://www.faa.gov/documentLibrary/media/Advisory_Circular/AC_23_1309-1E.pdf

[7] A. Agogino, G. Brat, D. Gopinath, Y. He, D. Hulse, L. Irshad, A. Katis, R. Lipkis, A. Mavridou, G. Pai, C. Pasareanu, I. Perez, T. Pressburget, and J. Schumann, “Recommendations on Evidence and Process for Certification of Learning Enabled Components in Aerospace Systems,” NASA Ames Research Center, Technical Report NASA/TM-20240006865, May 2024. [Online]. Available: https://ntrs.nasa.gov/citations/20240006865

[8] K. P. Murphy, *Probabilistic Machine Learning: An Introduction*. MIT Press, 2022.

[9] F. Kaakai, S. Adibhatla, G. Pai, and E. Escorihuela, “Data-centric operational design domain characterization for machine learning-based aeronautical products,” in *Computer Safety, Reliability, and Security. SAFECOMP 2023*, ser. Lecture Notes in Computer Science (LNCS), J. Guiochet, S. Tonetta, and F. Bitsch, Eds., vol. 14181. Springer, 2023, pp. 227–242. doi: 10.1007/978-3-031-40923-3_17

[10] T. Mitchell, *Machine Learning*. McGraw Hill, 1997.

[11] Y. Dong, W. Huang, V. Bharti, V. Cox, A. Banks, S. Wang, X. Zhao, S. Schewe, and X. Huang, “Reliability Assessment and Safety Arguments for Machine Learning Components in System Assurance,” *ACM Transactions on Embedded Computing Systems*, vol. 22, no. 3, Apr. 2023. doi: 10.1145/3570918

[12] M. Scheerer, M. Take, and J. Klamroth, “Quantifying Lower Reliability Bounds of Deep Neural Networks,” in *2024 IEEE 35th International Symposium on Software Reliability Engineering Workshops (ISSREW)*, Oct. 2024, pp. 247–254. doi: 10.1109/ISSREW63542.2024.00087

[13] S-18, Aircraft And System Development And Safety Assessment Committee, “Guidelines for Development of Civil Aircraft and Systems,” Aerospace Recommended Practice ARP4754 Rev. B, SAE International, Dec. 2023.

[14] S. Burton, L. Gauerhof, B. B. Sethy, I. Habli, and R. Hawkins, “Confidence Arguments for Evidence of Performance in Machine Learning for Highly Automated Driving Functions,” in *Computer Safety, Reliability, and Security. SAFECOMP 2019*, ser. Lecture Notes in Computer Science (LNCS), A. Romanovsky, E. Troubitsyna, I. Gashi, E. Schoitsch, and F. Bitsch, Eds., vol. 11699. Springer, 2019, pp. 365–377. doi: 10.1007/978-3-030-26250-1_30

[15] S. Burton, B. Herd, and J.-V. Zacchi, “Uncertainty-Aware Evaluation of Quantitative ML Safety Requirements,” in *Computer Safety, Reliability, and Security. SAFECOMP 2024 Workshops*, ser. Lecture Notes in Computer Science (LNCS), A. Ceccarelli, M. Trapp, A. Bondavalli, E. Schoitsch, B. Gallina, and F. Bitsch, Eds., vol. 14989. Springer, 2024, pp. 391–404. doi: 10.1007/978-3-031-68738-9_31

[16] L. Gauerhof, R. Hawkins, C. Picardi, C. Paterson, Y. Hagiwara, and I. Habli, “Assuring the Safety of Machine Learning for Pedestrian Detection at Crossings,” in *Computer Safety, Reliability, and Security. SAFECOMP 2020*, ser. Lecture Notes in Computer Science (LNCS), A. Casimiro, F. Ortmeier, F. Bitsch, and P. Ferreira, Eds., vol. 12234. Springer, 2020, pp. 197–212. doi: 10.1007/978-3-030-54549-9_13

[17] L. Gauerhof, P. Munk, and S. Burton, “Structuring Validation Targets of a Machine Learning Function Applied to Automated Driving,” in *Computer Safety, Reliability, and Security. SAFECOMP 2018.*, ser. Lecture Notes in Computer Science (LNCS), B. Gallina, A. Skavhaug, and F. Bitsch, Eds., vol. 11093. Springer, 2018, pp. 45–58. doi: 10.1007/978-3-319-99130-6_4

[18] S. Burton and B. Herd, “Addressing Uncertainty in the Safety Assurance of Machine Learning,” *Frontiers in Computer Science*, vol. 5, April 2023. doi: 10.3389/fcomp.2023.1132580

[19] B. C. Hu, L. Marsso, K. Czarnecki, R. Salay, H. Shen, and M. Chechik, “If a Human Can See It, So Should Your System: Reliability Requirements for Machine Vision Components,” in *2022 IEEE/ACM 44th International Conference on Software Engineering (ICSE)*, 2022, pp. 1145–1156. doi: 10.1145/3510003.3510109

[20] S. S. Roudposhti and R. D. Hawkins, “Specifying Safety Requirements for Machine Learning Components in Autonomous Systems: A Survey,” in *Proceedings of the 33rd Safety Critical Systems Symposium (SSS ’25)*, M. Parsons, Ed., February 2025. [Online]. Available: https://eprints.whiterose.ac.uk/222477/

[21] K. Dmitriev and G. Pai, “SAE G-34 AND EUROCAE WG-114 Joint Use Case Initiative with FAA,” FAA Artificial Intelligence Safety Assurance: Roadmap and Technical Exchange Meeting, November 2024. [Online]. Available: https://na.eventscloud.com/ereg/inactive.php?eventid=768017