The Importance of a System-Level Approach when Bringing in New Technologies in Avionics

Håkan Forsberg
School of Design, Innovation and Engineering
Mälardalen University
Västerås Sweden
hakan.forsberg@mdu.se

Kristina Forsberg
Saab Surveillance
Saab AB
Huskvarna, Sweden
kristina.forsberg@saabgroup.com

Joakim Lindén
Saab Aeronautics
Saab AB
Järfälla, Sweden
joakim.lindén@saabgroup.com

***Abstract*—In the era of multiple industry trends and new technologies, avionics systems can benefit from several innovations. The complexity of modern electronics is increasing quickly and is being introduced as never before in new applications. At the algorithm level, the use of deep neural networks helps to solve problems that were never believed to be doable before. At the architecture level, hardware artificial intelligence accelerators, embedded graphical processing units, embedded sensors, etc., make it possible to create very powerful new functions. The list of new technologies is long. Besides technical challenges, system integrity and availability must be assured when integrating these new technologies into avionics functions. In this paper, we present emerging technologies and why a system-level approach is necessary when implementing these technologies. We also introduce supporting means for design assurance and fault-tolerance techniques. We illustrate the importance of a system-level approach through an example. Our example shows that when developing functions with new technologies and fault-tolerant architectures, the system safety assessment process is crucial for properly implementing a fail-safe design. It is also challenging due to potentially new failure modes.**

Keywords—system safety, safety assessment, new technology, machine learning

I. INTRODUCTION

When new technology is introduced, the focus is on the functional level: what can the technology solve? The system is built around the function rather than focusing on the potential hazards connected to malfunction. If a proper top-down approach is used as described by the system safety process guidelines, the new technology may be used differently than planned for but be better suited to fulfill the required safety-requirements. Focus is also on trying to solve all tasks with a single solution, resulting in suboptimal results.

If, for instance, a convolutional neural network (CNN) is selected for object detection and classification, the focus is on solving these two tasks. An already pre-trained network is naturally selected from the "best available" networks as the base network. The pre-trained network is then fine-tuned on a specific task until accurate enough for the intended function. While this approach works for most systems, this may not be the case for safety-critical functions. Introducing new technology into safety-critical systems poses challenges to ensure reliability, integrity and availability. To illustrate this, imagine we develop a safety-critical system employing neural networks (NNs). In such system, it is crucial for decisions to be explainable to ensure integrity. At the system level, especially in object detection systems, explainability can be achieved through validation data and statistical analysis across all output classes [1]. The probabilities for correctly classifying objects should significantly outweigh those for other classes, and the identified bounding boxes must closely align with the ground truth. These statistics must conform to system requirements. To achieve high integrity, to mitigate false negatives (undetected objects) and false positives (incorrectly identified objects), diverse, redundant, parallel deep neural networks (DNNs) can be deployed [2].

A solid understanding of the system and intended functionality is also required when selecting the training dataset for NNs. There must be an assurance that the data used for training a model appropriately spans the operating design domain when operating in the real-world environment. Scenario diversity with scene-altering parameters like locations, daylight conditions, and weather conditions need to be covered [3]. To accelerate scene diversity and include cases not achievable in the real environment (e.g., testing obstacle detection on runways), and to lower the cost for training, synthetic datasets are the key to achieving a high level of integrity [4]. After training and validation of the NNs, system properties must be ensured when transitioning from the learning to the inference environment [1]. In the learning environment, the NN model is trained to behave as intended. The model is then transferred to the final hardware (inference environment) ready to be developed for use in the real application. Rarely is it the case that the same hardware is used for the two environments. The complex hardware in the inference environment draws less power and uses specific artificial Intelligence (AI) accelerators with limited computational accuracy (and sometimes with errors allowed). At the same time, this hardware must be more time-deterministic than that which is used in the training phase.

The current trend is also to integrate more and diverse computing cores in the same chip. Each computing core is developed for its specific purpose to accelerate the computations. Merging functions onto a single or a few chips may lead to more complex common mode analysis but may in some cases improve the detection of design errors through the inherent hardware diversity used.

In addition to all the new technology, advancements in supporting methods for safety analysis are emerging. These include not only the model-based safety analysis described in SAE ARP4761A [5] but also other techniques, such as newly invented deductive fault analysis methods, e.g., [6] or argument-based assurance and certification methods [7]. The latter is typically discussed for software but has been introduced for complex electronics as well [8,9]. Assurance cases provide assurance for a system through arguments justifying claims about the system. This can, in principle,

This work is partially funded by the Swedish Knowledge Foundation within the RELIANT Research School.

allow assurance cases to be better tuned to specific conditions of a system, and is therefore more agile than traditional assurance guidelines in adapting to new technologies and applications [7]. These innovative approaches can complement the well-established assurance methods and facilitate the adoption of new technology in safety-critical systems.

In this paper, we explain why a system-level approach is necessary when implementing new technologies. We do it through an example. We also introduce new fault-tolerance techniques that may assist in the integration of new technology.

The rest of this paper is structured as follows: Section II introduces the necessary background information. In Section III, we introduce an example system using new technology – a *Smart Eye* support for landing. In Section IV, we perform a safety assessment of our Smart Eye. In Section V, we discuss our approach and potential expansions of our system, and in Section VI we conclude the article.

II. Background

In this section, we begin with a short summary of system safety. We then introduce machine learning and approximate computing and highlight the difficulties of using these technologies in safety-critical systems. To support the implementation of the technologies, we also elaborate on emerging assurance methods, new safety assessment methods, and time-dependent knowledge graphs.

A. Guidelines for development of aircraft systems

Avionics systems development goes hand in hand with system safety and follows the guidance in SAE ARP4754B [10]. The safety assessment is carried out using guidelines from the document SAE ARP 4761A [5].

The code of Federal Regulations, Title 14, §25.1301 (in Europe CS 25.1301), requires each equipment installed on an aircraft to be appropriately designed for its intended function. Every limitation specific to that equipment must be taken care of. In addition, §25.1309, complements this requirement by considering any foreseeable operating conditions, i.e., no unintended functions are allowed. Thus, for any new technology installed, all limitations must be well-known, and the functionality and operating environment must be well understood.

Following RTCA/DO-178C and RTCA/DO-254 (EUROCAE ED-12C and ED-80), ensures development/design assurance of the underlying software and complex electronics. For new technology, specific assurance may be required, e.g., guidance for the use of NNs (algorithm level) in safety-critical applications.

B. Machine learning in safety-critical systems

Machine Learning (ML) has impressive abilities to approximate complex functions and is already used in safety-critical functions, e.g., self-driving cars. However, using ML in these systems raises concerns [11,12]. To be able to rely on ML in safety-critical applications, fail-safe design principles must be enforced. For example, we need integrity and quality to ensure intended function and prevent failures, and proven reliability so that multiple, independent failures are unlikely to occur at the same time (e.g., during one flight for aircraft).

The level of integrity is not well-known for systems using ML due to their inability to guarantee that the predicted outcome is correct. By using automated methods such as Neural Architecture Search (NAS) and AutoML, the process of finding the "best" architectures for decision making can be partially automated. However, these methods rely on the hypothesis that any neural network (NN) can be approximated adequately well by selecting the right subnetwork out of a large random network without mastering the finite but still combinatorial explosion of possible networks [13]. For embedded systems another problem arises: the trained model needs to be embedded in the real environment's hardware, which is typically not the same as the hardware used for training. Network pruning, quantization and other tricks are typically performed to make the model fit in the real environment. For dependable systems, the trained model's properties (learning environment) need to be transferred to the real-world model (inference environment) and this transformation must be guaranteed. In future work, we aim to explore a mathematical approach for ML and specifically for the transformation between the different environments. With the theory of submodular functions, it is expected that certain theoretical and practical problems arising in ML can be solved [13].

When introducing ML in a system, other new technologies may be required for reliability or other system properties. To give an example, Schorn et al. [14] introduce a triple modular redundant feed-forward neural network to detect anomalies (in this case soft errors) in a larger DNN. This complexity further promotes a system-level approach. It is at the system level that monitors are decided to ensure safety (while monitors for reliability can be set at lower levels like in the case above). For an overview of error-mitigation techniques for DNNs, see Mittal [15].

By using fault-tolerant design principles (architectural mitigation and diversity) for multiple parallel NNs, uncertainty can be better estimated and reduced compared to single networks. One example is deep ensembles [16-18]. In a deep ensemble, several networks try to identify the same objects individually. If, for instance, a majority of the networks identify runway lights with high probability and every other object with low probabilities, the ensemble's statistics clearly indicate it is a runway light. The diversity between the networks in an ensemble can be applied in the training process with different training sets and/or epochs, or by using different architectures. With the human-in-the-loop, the level of uncertainty can be further reduced through smart guidance during training of deep ensembles. The statistics for classification of output classes are enhanced (improved explainability). The strategy for selecting the ensemble of networks has evolved over time and several methods and fusion strategies have been suggested [17]. We believe specific mathematical functions may be used for diversity measures between NNs in deep ensembles. In addition, in the future, deep ensembles may be embedded in the real environment, opening for other diversity actions, e.g., different pruning or quantization, with the properties of the ensemble still being guaranteed. Network pruning or quantization is typically used in DNNs to reduce the size of

the network, increase the speed or to overcome certain overfitting problems.

Another important aspect concerns the development assurance of machine-learning based systems. For these systems, the functionality is heavily dependent on input training data. Traditional software (SW) and hardware (HW) development guidelines cannot satisfy the required design assurance in the training phase. Other types of design assurance are needed. This was noticed early in the avionics industry. Concepts for design assurance of NNs have been introduced [1,19] in conjunction with an AI roadmap [20]. The Aerospace Vehicles Systems Institute and the main standardization organizations are working towards assurance guidelines for the use of ML in avionics as well [21-24]. In this article, we assume ML design assurance guidelines to be followed on the algorithmic level for development and integration of systems reliant on ML.

C. *Approximate computing*

Approximate computing is another new technology where computation accuracy is traded off (errors are allowed to happen in the computations) for better performance or lower energy consumption [25]. Specifically, the latter is used as an argument for bringing greenness to computing [26]. Approximate computing is typically used in certain applications where specific types of errors can be acceptable from a system requirements perspective, e.g., minor flaws in images. Approximate computing can be performed in multiple ways and at different levels, e.g., at the logic level, at the arithmetic circuits level, or at the system level including systems using AI [27]. For critical-data, approximate computing should be avoided. Only if the traded accuracy is deterministic (may be statistically), integrity can be controlled. It shall be noted that Google in their tensor processing units use approximate computing to reduce power consumption and IBM applies it in their on-chip AI accelerators [27]. Thus, a system designer implementing AI based systems for safety-critical applications needs to know if approximate computing is already built-in in certain functionality (hardware) and if it may have potential impact on the application. Approximate computing shall not be mixed with network pruning or quantization.

D. *Supporting COTS hardware assurance methods*

When introducing new technology in safety-critical systems, there must be convincing arguments that the new technology does not violate the safety requirements (or other requirements as well). To assist in this process, supportive assurance methods can be used. One such method is argument-based assurance [7]. However, care must be taken so that the evidencing part ensures a rational inquiry in the real world [28]. For new technology, such as embedded systems using DNNs, we have shown [29] the early steps to be taken using a generic assurance approach to get the needed flexibility in the way we argue that the COTS assurance objectives are met. These early steps may be used for the introduction of any kind of new technology.

E. *Model-based system safety analysis*

In the recently revised document "Guidelines for Conducting the Safety Assessment Process on Civil Aircraft, Systems, and Equipment," SAE ARP 4761A [5] a new safety assessment method is introduced, the model-based safety analysis (MBSA). This method shall not be mixed with traditional model-based systems engineering. MBSA is said. to be capable of replacing fault-tree analysis (FTA), Markov analysis, or dependency diagrams, and may help in other analysis methods, such as common mode analysis and particular risk analysis. MBSA supports the safety analysis of a function, addresses complex functions, and facilitates the communication between system and system safety engineers through models [5]. The safety analysis is performed through Failure Propagation Models (FPMs), as seen in Fig. 1. Design documents include system description, architecture diagrams and requirements.

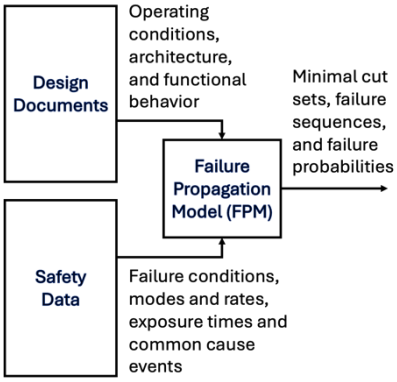

Fig. 1. MBSA method (*figure based on Fig. N2 in [5]*). The failure propagation model (FPM) represents the system architecture including its dysfunctional behavior. The FPM gets input from design documents and safety data and generates minimal cut sets, failure sequences, and failure probabilities.

We believe MBSA can support the use of new technology in safety-critical systems. Typically, novel solutions change more frequently, and new failure conditions (FCs) may show up due to unfamiliar behavior, and these must be taken care of. Also, multiple failure conditions may occur. The FPM is ideal for handling these situations. Once the architecture diagrams and requirements are stable, we still believe the use of traditional FTAs should be employed (at least for communication with customers and certification authorities).

F. *Time-dependent solutions based on knowledge graphs*

To support structured management and propagation of characteristic time series information, Graß et al. [30], introduce the concept of knowledge graphs for time series data, also known as temporal knowledge graphs (TKGs). Their solution relies on automated knowledge discovery and machine learning.

TKGs can be used for context recording from continuous tracking of sensor readings. Imagine we use a DNN to track multiple objects from camera image streams. TKGs can then be used to build graphs over identified entities. We assume TKGs can be used for anomaly detection, i.e., deviations from expected patterns from the sensor readings. It could be something like:

> *An airplane (Airplane_1) detects a new light (Light_A) at a location (Point_Y) close to airport (Airport_Z) at a specific time (2024-06-25T112:12)*

Knowledge discovery in databases (KDD) is a field within Knowledge Graphs concerned with finding and

sorting out relevant data from a TKG, whereas the TKG offers a structured representation of the collected data, including these discoveries. This structured storing of aggregated information makes it accessible for other parts of the system, at the appropriate abstraction level, such as a monitor function tasked with detecting sensor drift or anomalous sensor readings. This way of aggregating system status information is indeed a methodology to structure the abundance of sensor data available in most systems today in a way which lends itself nicely to other dynamic techniques such as recommender systems, predictive maintenance and other data-intensive tasks.

When the information is static, it is stored in a Static Knowledge Graph (SKG). With SKGs, we have prior knowledge to which we may relate our sensor inputs. When presented to the KDD function, static information is already available before takeoff (e.g. layout of markers, light fittings at a particular runway, runway material etc).

Another application of TKGs, could be to detect anomalies in the NNs themselves, for example after single event upsets (like Schorn et al. [14] did with their triple-modular redundant NNs for detecting abnormalities in deeper NNs). This would be possible since the TKGs continuously stores the generated outputs from the NNs.

III. Example - Smart Eye Support for Landing

The FAA revised the rule for the use of enhanced flight vision systems (EFVS) and pilot compartment view requirements in December 2016. The revised rule permits the pilot to use EFVS instead of natural vision to continue descending below 100 feet to touch down and rollout under certain conditions [31]. Using EFVS equipment onboard for landing requires less ground infrastructure at the airport (CAT II and III equipment). Thus, landing in hazy weather conditions may be performed on many more and less equipped airports.

When the pilot's eye can be enhanced and temporarily replaced with a *Sensor Eye* during low visibility conditions, improved operational weather minimums can be achieved. Weather minimums ensure that pilots have sufficient visibility and spatial orientation to navigate safely, both in the air and near the ground, during takeoff, enroute, and landing.

To support the pilot, we suggest using a dedicated *Smart Eye* to identify lights, surfaces and markers, for making descent decisions. The Smart Eye is composed of new technology. Fig. 2 shows our proposed architecture without redundancy. We assume that the Smart Eye is part of an enhanced EFVS (EEFVS) system.

Pilots using an EFVS for landing must be careful not to conclude that there are no obstacles in the flight path just because the enhanced images don't show any obstacles. Also, in worst case scenarios where no published vertical guidance exists, and the pilot must trust flight path (FP) vectors and FP angle reference cues, obstacles may appear in the real world but not in the EFVS [31]. Obstacle detection must be in place from decision altitude (DA) / decision height (DH) and all the way down to landing and rollout (or in case of a missed approach after DA/DH). In addition, if millimeter wave radar is used to detect obstacles, cluttering may appear, misleading the pilot to believe there are false obstacles.

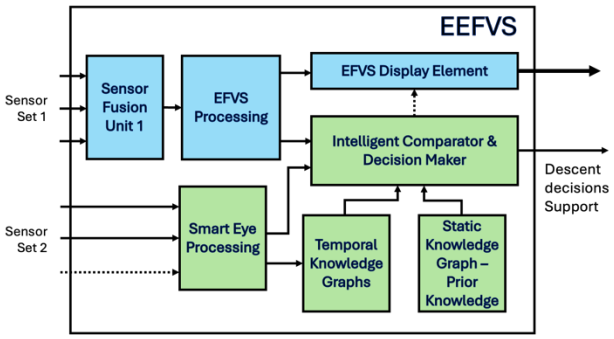

Fig. 2. The suggested implementation of the example system. Light blue boxes are part of the conventional EFVS while the light green boxes are new technology and part of the enhanced EFVS with a Smart Eye. Separate and diverse sensor data is sent to the Smart Eye, which detects lights, surfaces, and markers. The detected objects are stored and processed with Temporal Knowledge Graphs (TKGs) and compared with static information, e.g., airport reference objects and runways, located in a Static Knowledge Graph (SKG). The intelligent comparator and decision maker reads data from the SKG and TKGs and from the EFVS processing unit and informs the pilot of which reference objects have been detected and where.

At the same time as the pilot pays attention to obstacles, he/she needs to pay attention to detect reference objects through the EFVS to descend below DA/DH and later below 100 feet above the touchdown zone elevation (TDZE[1]).

A. *The EEFVS system*

The EEFVS system is assumed to operate under the condition for the purpose of research and development and is assumed to comply with all applicable EFVS requirements. The EFVS part of the EEFVS (blue boxes in Fig. 2) has imaging sensors that display the forward imaging scene. There are several types of sensors that can be used, such as forward looking infrared (FLIR) cameras (may use different infrared spectrums), low-light level image amplifier (LLIA), millimeter wave radar, or millimeter wave radiometry. There are several reasons for using diverse types of imaging sensors, one being independence requirements, and others being the incapabilities of certain sensors, e.g., FLIRs may not detect lights from LED-based lamps. In our system, see Fig. 2, we assume Sensor set 1 consists of one ordinary electro-optical camera, one millimeter wave radar, and one infrared sensor independent from the sensors in Sensor set 2. Sensor set 2 consists of a shortwave FLIR and an LLIA. In our example, we consider the most critical landing, i.e., EFVS operations to touchdown and rollout, see Fig. 3.

B. *Requirements for decision to descend below DA/DH*

The requirement for detecting reference objects through the EFVS system for decision to descend below DA/DH is either:

1. The approach light system

or both of the following:

2. The runway threshold
3. The touchdown zone (TDZ)

The runway threshold can be detected through either a) the beginning of the runway landing surface, b) threshold lights, or c) runway end identifier lights. The touchdown zone can

[1] Elevation here refers to the highest elevation on the runway between 0 to 3 000 feet into the landing surface.

be detected through either a) runway TDZ landing surface, b) TDZ lights, c) TDZ markers or d) runway lights.

C. *Requirements for decision to descend below 100 feet above TDZE*

The requirement for identifying reference objects 100 feet above TDZE to descend even further, see Fig. 3, is to detect one of the following four reference objects:

1. The runway threshold,
2. The lights or markings of the threshold,
3. The runway TDZ landing surface, or
4. The lights or markings of the TDZ

The detection can be done through visual reference or through EFVS. In our example the detection is through the Smart Eye.

D. *EFVS operations requirements to touchdown and rollout*

To be able to carry out EFVS operations in lieu of natural vision from DA/DH down to landing and rollout, several requirements must be followed. The EFVS must display important aircraft flight information (see [31], §4.1.1.3) and many parameters must be aligned and scaled with the external view (they must be conformal). Additional requirements also apply including obstacle detection. In our example, the Smart Eye is not affected by the above requirements.

When two or more pilots are required, a pilot monitoring function showing the pilot's flying EFVS sensor imagery must be present. EFVS operations to touchdown and rollout must be capable of handling any failure of any component in the system.

E. *Smart Eye function using new technology*

The whole idea with the Smart Eye is to support the pilot with decisions to descend below DA/DH and below 100 feet above TDZE when using the EFVS, to offload his or her burden of many other things during this busy landing moment. Once the Smart Eye detects the required reference objects, the pilot gets information about which reference objects are detected and where they were found (via bounding boxes), to support the decision to descend even further.

Since we are dealing with detecting reference objects, it is tempting to let the Smart Eye act as an obstacle detector as well. However, this complicates the use case and has therefore not been included (see the discussion section for additional information).

Our Smart Eye starts to detect reference objects before DA/DH and continues to do so all the way down to landing. At two heights, DA/DH and 100 feet above TDZE, see Fig. 3, decisions to continue to descend are taken based on detected objects. The following reference objects shall be detected:

1. Approach light system
2. Beginning of the runway landing surface
3. Threshold lights
4. Runway end identifier lights
5. Runway TDZ landing surface
6. TDZ lights
7. TDZ markers
8. Runway lights
9. Markings of the threshold

If at least one of the following elements (a single object or a pair of objects) in the list below (reference objects according to the numbered list above) is detected and presented by the Smart Eye just before reaching DA/DH decision point, the pilot is informed and can descend below DA/DH:

{(1), (2,5), (2,6), (2,7), (2,8), (3,5), (3,6), (3,7), (3,8), (4,5), (4,6), (4,7), (4,8)}

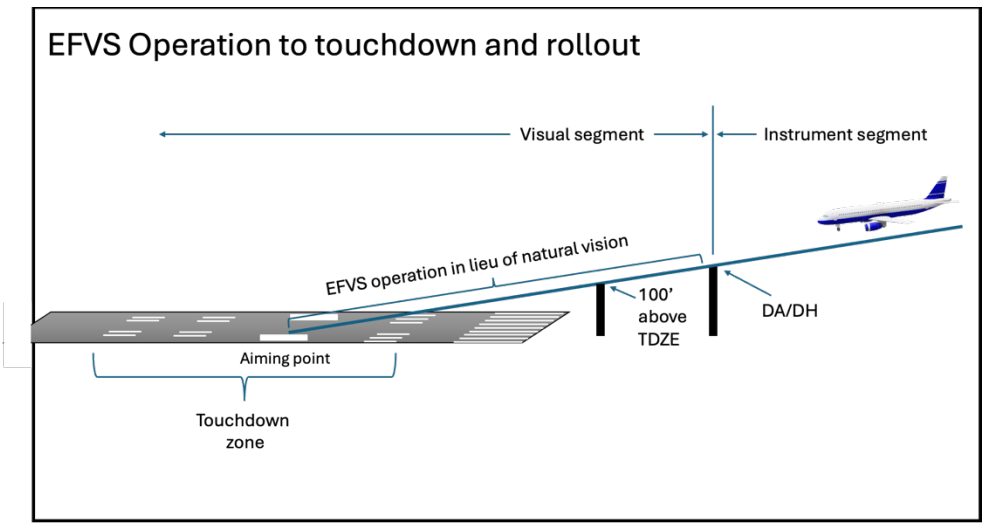

Fig. 3. Concept of EFVS operation to touchdown and rollout. The aircraft approaches landing in the instrument segment. To proceed below DA/DH, certain reference objects must be detected. In the visual segment, the complete landing is performed with EFVS operation in lieu of natural vision. When reaching 100 feet above the touchdown zone elevation (TDZE) other reference objects must be detected before descending further.

If one of the following elements of reference objects in the list below (numbers according to the list above) is detected and presented by the Smart Eye just before 100 feet above TDZE decision point, the pilot is informed and can continue to descend for landing and rollout.

{2, 3, 4, 5, 6, 7, 9}

While it is tempting to train and let a single convolutional neural network (CNN) be the Smart Eye which identifies all nine objects above, it should be noted that there are differences in how the different objects are detected. Our Smart Eye shall detect three types of objects: surfaces, markers and lights. The latter consists of detecting five different types of lights, which may not be detected clearly by FLIR sensors but probably much better by the LLIA sensor. The fact that the reference objects to be detected are sensor-sensitive, different detection solutions may be required.

IV. Safety Assessment of the Smart-Eye

The acceptable means to comply with certification regulations is to follow SAE ARP4754B when developing avionics systems. This guidance document recommends a top-down requirements driven development starting with the Functional Hazard Assessment (FHA) performed for all A/C level functions.

Here we present a limited part of the established safety assessment process for the example EEFVS system, pictured in Fig. 2. The assessment process includes the Descent Decision Support part (green boxes in Fig. 2) with the aim to elaborate on possible shortcomings when applied for new technologies. The integrated EFVS part (blue boxes in Fig. 2) is not included.

A. System FHA – Smart Eye

A functional hazard assessment (FHA) identifies the failure conditions for all functions, analyzes the effects on the aircraft and flight crew, and assigns the corresponding criticality classification (No Safety Effect, Minor, Major, Hazardous, Catastrophic) while considering both loss of functions, and malfunctions. Table 1 shows our Smart Eye FHA considering failure conditions in terms of integrity and availability of function. The Smart Eye functionality informs and displays visible reference objects to the pilot during approach until touchdown and rollout.

Table 1. FHA for the Smart Eye

#	Failure Condition (FC)	Effects	Classif.
1	No identified visible reference object when reaching DA/DH	Pilot executes go-around	MIN
2	Loss of reference object during guidance (below DA/DH)	Pilot executes go-around	MAJ
3	Loss of reference object during guidance (below 100 ft above TDZE)	Pilot executes go-around	HAZ
4	Identified reference objects tied to wrong static reference objects	Could result in hard landing or controlled flight into terrain. Pilot has no means to detect the error and will follow the guidance.	CAT

The FHA identifies the hazard levels associated with the Smart Eye failure conditions to determine the required system design assurance and safety levels. The FHA derived safety requirements are:

SR1. Loss of reference object above DA/DH shall have the probability of less than $1.0\ 10^{-3}$ /FH
SR2. Loss of reference object below DA/DH shall have the probability of less than $1.0\ 10^{-5}$ /FH
SR3. Loss of reference object below 100 ft above TDZE shall have the probability of less than $1.0\ 10^{-7}$ /FH
SR4. Erroneous Reference Object for Descent Decision Support presented to pilot shall have the probability of less than $1.0\ 10^{-9}$ /FH
SR5. Smart Eye system Development Assurance Level (DAL) shall be A (CAT FC requires DAL A)
SR6. No single fault shall lead to Erroneous Reference Object for Descent Decision Support presented to pilot

B. PSSA – Smart Eye

The preliminary system safety assessment (PSSA) is a top-down approach evaluating how the proposed system architecture can meet the safety objectives resulting from the FHA. The PSSA process decomposes and allocates FHA safety requirements and determines derived safety requirements for all design items. In this PSSA example, two conceptual fault trees are constructed for the Smart Eye system (green boxes in Fig. 2). One representing *Loss of* and one representing *Malfunction*.

The Loss of tree, Fig. 4, includes failure rates. It shows a possible allocation of failure rates to evaluate if suggested architecture can meet required availability budget from the FHA. All failure rates are fictive and included for redundancy discussion regarding availability.

The Malfunction tree, Fig. 5, does not include any failure rates, the tree shows how the design intend to meet the required Integrity level from FHA and identifies independence requirements.

1) Fault tree analysis

The FTA TOP 1 – Loss of reference object (pilot guidance) is constructed with three branches, see Fig. 4. HW faults contributing to the TOP FC in the leftmost branch represent Smart Eye input equipment necessary to perform the function Identify Reference Objects. The middle branch includes the NN not detecting any reference objects. HW faults contributing to the TOP FC in the right branch represent Smart Eye platform HW, i.e., power, processing capability, Input/Output interfaces (I/O).

The FTA TOP 2 – Reference object tied to wrong static reference object is shown in Fig. 5. The Smart Eye is designed to achieve high integrity through the possibility to cross-check the NN's identified reference objects with known reference objects (markers, lights and specific surfaces for each runway) preloaded in a static database. TOP 2 illustrates this design, where the left branch under the AND-gate represents the new technology part. The right branch is the static reference database used by the cross-check monitor.

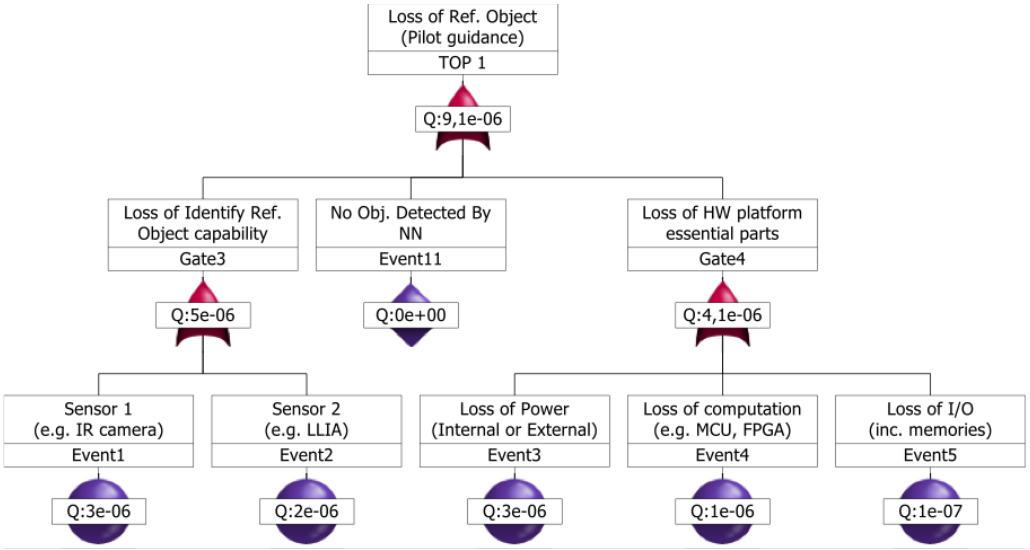

Fig. 4. FTA TOP 1 - Loss of reference object, pilot guidance. Q is the failure frequency per flight hour (normally indicated by λ).

2) PSSA results

The proposed architecture does not include redundancy for availability and from TOP 1 it is seen that the safety requirements SR1 and SR2 are met while SR3 is not met. Even though failure rates are fictive in this example it is not realistic to meet hazardous FCs without redundancy.

Design assumption: The processing part of the Smart Eye which uses NNs to detect and classify objects, also calculates location data for the detected objects. (This is possible through the knowledge of the plane's horizontal position, altitude, attitude, roll and yaw plus the fact that it is only objects on ground that are detected.)

Independence requirement: NN location data stored in the TKGs has a dissimilar source than the SKG's location data. SKG's location data are the exact positions of the static objects.

C. Fail-safe design

For neural network (NN) design assurance on the algorithm level, we suggest following the W-assurance model described in [1]. Once the standardization organizations finalize their guidelines, these should be followed. Following such guidance documents may also reduce the possibility for adversarial attacks (inputs to the machine learning model designed by an attacker to maximize the model making mistakes).

The Smart Eye uses new technology in the form of NNs. To select the best possible NN for detecting the reference objects including lights (LED-based as well as normal lights) in hazy weather conditions and with the option of two sensor inputs (SW-FLIR & LLIA), will be a crucial task to perform. Research in detecting objects in bad weather conditions is mainly focused on detecting pedestrians in front of autonomous cars. Typically, forward radar, normal camera and LiDAR are used. Reusing these ideas for our system may be limited.

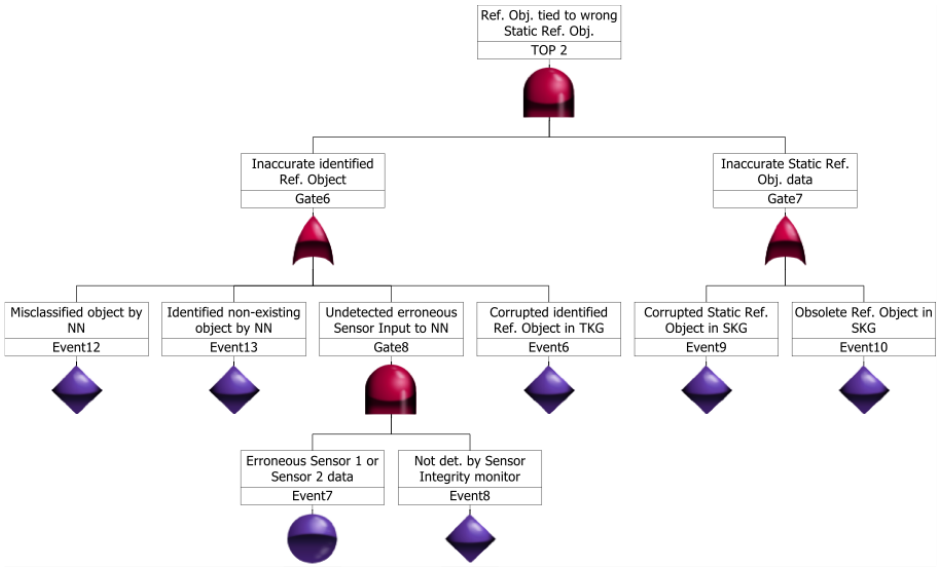

Fig. 5. FTA TOP 2 - Reference object tied to wrong static reference object.

While most NNs used for object detection have been trained on "sunny days" images, and are supposed to work normally in clear weather, our system will never be used in those conditions (clear days or clear nights). Our system works normally in worst-case conditions with heavy fog or rain, or low-altitude clouds. Thus, the operating design domain (ODD) is different and training data should be focused for those conditions. Then, different sensors see differently through different weather types. Therefore, the selection of correct sensor types is important and the preparation of the datasets for training needs to be carefully planned.

In the FTA TOP 1, it is shown the NN may not detect any reference objects despite the sensors are okey. This results in a go-around. For an NN to completely ignore all detection of objects, either the training has been extremely poor, or the ODD has not been clearly identified. The former is not realistic, but the latter could happen. Poor training resulting in the NN missing a few objects may not result in a go-around since other objects will be identified. Poor training resulting in incorrect object detection (including false detections of non-existing objects), will be noticed by the system, see FTA TOP 2. These objects are not shown for the pilot but instead ignored. However, other reference objects must still be correctly detected. Thus, the focus should be on the ODD. Every bad weather condition possible during day to complete darkness in the night must be taken care of, suggesting the use of synthetic data for training. In addition, five different lights, two types of surfaces and two types of markers shall be detected and classified from a many of approach angles and inclinations. The produced synthetic data need to be near the real-world data. Any gap between the simulated and real-world data needs to be controlled. Perhaps 3D models of the airports including all lights need to be used. See Lindén et al. [4] for an implementation of 3D models in synthetic datasets.

To achieve higher integrity, two NNs can be used: one for surface and marker detection and another for light detection. These two NNs should be trained with diverse "biased" incoming sensor data. That is, the NN used for surface and marker detection should rely mostly on the FLIR sensor data and then suppress the LLIA sensor data. Light detection, on the other hand, should rely more on the LLIA sensor data than the FLIR sensor data. See Fig. 6 for a possible Smart Eye processing architecture.

If an "unknown" weather condition appears (NN not trained for this scenario), it is sufficient if one of the differently trained NNs with diverse biased sensor inputs works, to be able to achieve descent clearance as specified. This is true since at both decision heights, a decision to descend can be made from detection of lights only or from detection of the surface and markers only. However, the number of possible elements to make the decision from is reduced. This scenario might be considered as a degraded mode. If the sensors are truly independent and a normal descent decision can be taken with one sensor only, then the OR-gate between Sensor 1 (Event 1) and 2 (Event 2) in the FTA 1, see in Fig. 4, should be changed to an AND-gate.

To remove any inconsistencies and to ensure correct behavior from out-of-distribution data, guidance is given in [1] (Chapter 5 Safety Assessment). Additional guidance is given by Lindén et al [3]. They introduce metrics to quantify similarity, used to estimate how a model will perform on out-of-distribution data.

We suggest pilots and other experts to be in the loop to observe the NNs' outputs when validating the classified data at the end of the training phase.

In FTA TOP 2, we see that the pilot is only presented with invalid data if a reference object is inaccurately identified **AND** the corresponding data in the SKG is corrupt or obsolete. The left branch may happen if the NN identifies an incorrect reference object or a ghost-object (detected non-existing object), data is corrupt in TKG, or the sensor is incorrect and not detected by the sensor monitor. To overcome corrupt data in the TKG or SKG databases, sufficient data encryption shall be used. It is assumed the SKG database is regularly updated (e.g., every 28 days or so). Uncalibrated sensors or ageing sensors should be detected by the sensor monitors. Thus, the likelihood for FTA TOP 2 to happen is extremely low.

If an NN cannot be selected and trained to avoid multiple reference objects to be identified incorrectly or several ghost-objects to be identified, the availability of the system may become too low. Then, the use of multiple diverse parallel (MDP) networks may help. See Stepien et al. [2] for inspiration. Initial tests will reveal if MDP is necessary to use or not. Going even further and using deep ensembles may be used if generalization will be a problem [17]. The improved statistic from the ensemble helps in decisions to correctly classify the reference objects. It is, however, difficult to implement Deep Ensembles in the real hardware and the increased power envelope may constitute a problem.

As our system is built with comparison of static objects in an SKG, the likelihood to create false runways with false lights and markers, and making a pilote attempt landing on a fake place can be neglected.

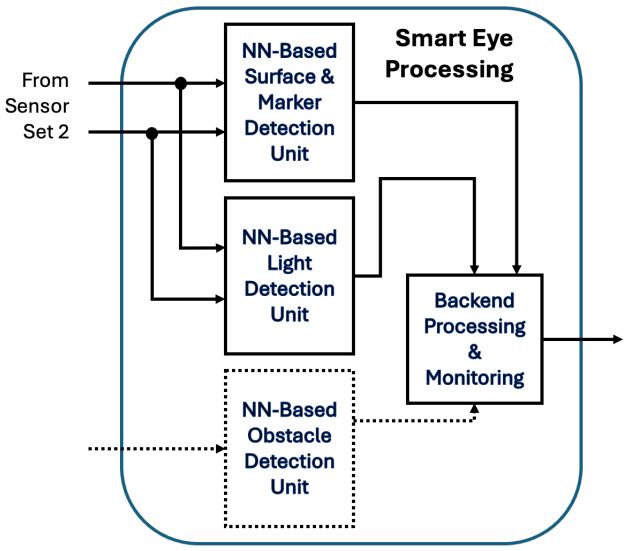

Fig. 6. One possible Smart Eye processing solution. The Surface & Marker Detection Unit identifies surfaces and markers, i.e., 2, 5, 7 and 9 in the list in Section III E above. The Light Detection Unit detects the five different types of lights, i.e., 1, 3, 4, 6, and 8 in the list. The Obstacle Detection Unit detects obstacles in the flight path. It is not part of the example but included in the discussion. The Backend Processing & Monitoring performs post-processing including local monitoring of the NNs.

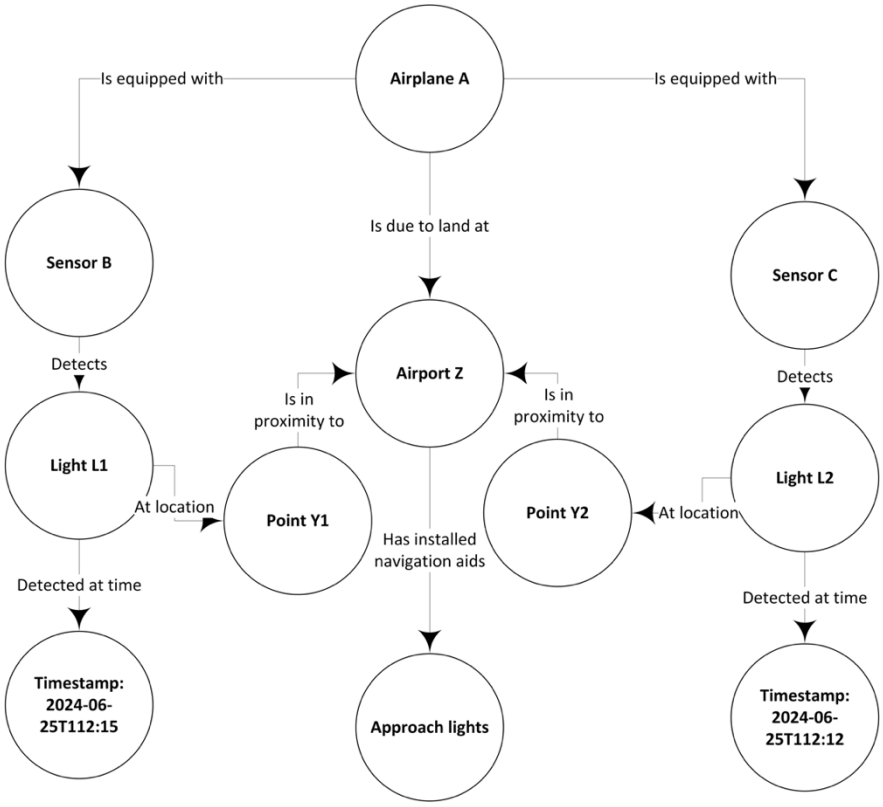

Fig. 7. Graph representation of gained knowledge in landing scenario. The graph structures the information in a way which makes it possible to deduce higher level intelligence from simpler sensor readings.

Smart guidance in the training with the human-in-the-loop may further reduce uncertainty and is recommended. The Surface and Marker Detection Unit should be implemented in a similar way as the light detection unit, however, with diverse bias of the sensor inputs.

To be able to fulfill the requirement that our system should work despite any failure of any component from DA/DH downto touchdown and rollout, we assume dual EEFVS to be implemented with one of them in hot standby.

To deduce higher level intelligence from the different sensors and the output from the neural networks in the Smart Eye, knowledge graphs will be used. See example over a landing scenario in Fig. 7. Besides identifying higher level information about objects and their movements, anomalies in the sensors and the DNNs may be identified with this technology. The knowledge graphs help to extract information in the different phases (descent below DA/DH and descent below 100 feet above TDZE). Also, since, the detection of different reference objects may render a go for descent below DA/DH and below 100 feet above TDZE, there is a need for a higher-level decision module like the Knowledge Discovery in Databases (KDD).

V. Discussion

In this section, we discuss our example architecture and a possible extension of it - an obstacle detector. In addition, we consider the use of new system safety methods and the use of complex COTS hardware.

A. Selection of the example architecture

In this article, we selected a use case based on an Enhanced Flight Vision System (EFVS), where landing in hazy weather is performed under EFVS operation in lieu of natural vision. The pilot flying uses a head-up display or a helmet with goggles, where the normal vision is enhanced with infrared camera images, radar-based images, or low-level light amplification or other means. Once the pilot detects certain reference objects with the enhanced vision, she/he is allowed to descend further (this concerns two different heights). At the same time, the pilot must ensure no obstacles are in the flight path. Obstacles may occur in the real-world in certain circumstances but may not be visible in the enhanced vision mode due to specific reasons. In our architecture, we introduce a Smart Eye to support the pilot with the task of identifying reference objects to support the descent decisions made by the pilot. Our Smart Eye solution is fictive and uses neural networks (NNs) solely for the purpose of showing the difficulties to implement new technology and why a system approach should be used. We also recognize the selection of input sensors to the EFVS and the Smart Eye from the eyes of a novice. The sensor selections should be carried out carefully, to achieve the best possible functionality.

We suggest the use of two different NNs using different sensor inputs and that detect different types of objects. There are many other possible solutions that could have been used. The first obvious one is to use a single NN detecting and classifying all nine different objects. The complete training time should then be focused on a single network possibly resulting in better detections of the objects. Another solution

most probably rendering higher integrity than a single network would be using two diverse NNs trained differently, but with the same goal, to detect and classify all nine objects. (The training time would have been split between the different networks.) The integrity increase would of course be hard to measure more than through validation tests. Why did we choose another approach? From the system level, we recognized that different sensor types detect light, surface and marker reference objects differently. We then realized that descent decisions can be based on light reference objects alone or on surface and markers alone. By using two NNs detecting different types of objects, we can emphasize the training of each of the networks differently (with diverse goals), with their respective sensor inputs and with the assumption that the detection task will be more robust and that even higher integrity can be achieved. We have, however, not validated our ideas yet on our Smart Eye example and can only speculate in the results.

B. Using the Smart Eye as an obstacle detector

In Fig. 6, an NN-based Obstacle Detection Unit is present (dashed module). This unit is not part of our example but is included here for discussion. The obstacle detection unit independently detects obstacles in the flight path. When using enhanced vision sensors that can "see through" clouds or heavy rain, other information such as colors may disappear and thus the pilot may potentially miss obstacles.

To detect obstacles with correct positions but without the need to correctly classify objects, and at the same time reduce both false positives and false negatives, diverse and parallel neural networks can be helpful [2]. The Obstacle Detection Unit may be implemented with three or more DNNs, diverse enough to reduce the number of undetected and ghost objects. Undetected objects may result in collisions and identified objects that do not exist may result in false abort landing scenarios. Furthermore, identifying the physical distance and location of any obstacle (including the ones without reference in the SKG) is harder than detecting distance and location to reference objects with well-known positions.

C. The use of new system safety analysis methods

We did not have time to test the new safety assessment method, MBSA, in SAE ARP 4761A for our example case. MBSA seems promising for systems with new technology due to novel solutions change more frequently. However, M. Sun et al. [32] state that one of the challenges using MBSA is to assure the adequacy of the fault models for newly designed components. Do we know the fault models for NNs accurately enough?

D. Implementing new complex COTS hardware

Argument-based assurance methods can be used to support COTS hardware assurance when implementing new technology but need further investigations for use in real projects using complex electronics such as embedded AI accelerators.

Advanced heterogeneous computing cores have been used in the car industry for autonomous driving for many years now. Tesla introduced their FSD computer for autonomous driving already in 2019 [33]. However, the regulations for avionics is different. To have better control over the hardware used, AI accelerators (without approximate computing algorithms) may be implemented on FPGAs (with own developed code or soft IP cores). To fit the trained models on the real hardware, the NNs may need to be pruned, and the accelerators may work with quantized data. Careful testing of the end system is required (and an essential part in the W-assurance model for NNs [1]) to ensure that system properties from the training phase are maintained in the final system using real hardware.

VI. CONCLUSIONS

In this paper, we elaborate on the significance of using a top-down system development approach when implementing new technologies in avionics and conducting system safety work following the guidance in SAE ARP4761A. We introduce an example: a Smart Eye using neural networks (NNs) to identify and classify reference objects to support pilots during landings in hazy weather conditions. We then perform a safety assessment of the Smart-eye and propose a proper fail-safe architecture.

We consider the main challenges when integrating new technologies to be process assurance, understanding new failure modes, and determining how to limit or divide the functionality into manageable parts.

The main conclusions are that we suggest using knowledge graphs in conjunction with NNs to organize and aggregate sensor information efficiently, and to use a system development approach that includes a proper safety assessment when new technologies are introduced into safety-critical applications. We also conclude that it is possible to address the use of NNs with conventional safety assessment processes, i.e., FHA, PSSA and SSA, but common mode analysis (CMA) to assure independence is challenging. In classic CMA, independence is used to achieve integrity and for high complexity diversity might be required. For NNs, the situation is more complicated. Two diverse NNs trained differently but with the same goal, may still lead to the same false positives and negatives.

REFERENCES

[1] EASA and Daedalean, "Concepts of design assurance for neural networks (CoDANN) II," May 2021.

[2] H. Stepien, M. Bilger, H. Forsberg, B. Lindgren, and J. Hjorth "A novel method for detecting UAVs using parallel neural networks with re-inference," 33rd Congress of the International Council of the Aeronautical Sciences (ICAS 2022), Sept. 4-9, 2022.

[3] J. Lindén, H. Forsberg, M. Daneshtalab, and I. Söderquist, "Evaluating the robustness of ML models to out-of-distribution data through similarity analysis," In: Abelló, A., et al. New Trends in Database and Information Systems. ADBIS 2023. Communications in Computer and Information Science, vol 1850. Springer, Cham. https://doi.org/10.1007/978-3-031-42941-5_30

[4] J. Lindén, G. Burresi, H. Forsberg. M. Daneshtalab, and I. Söderquist, "Enhancing drone surveillance with NeRF: Real-world applications and simulated environments," AIAA/IEEE 43rd Digital Avionics Systems Conference (DASC). IEEE, 2024, in press.

[5] SAE ARP4761A, "Guidelines for conducting the safety assessment process on civil aircraft, systems, and equipment," Revised Dec. 2023.

[6] C. Vitucci, T. Westerbäck, D. Sundmark, H. Forsberg, and T. Nolte, "A deductive fault analysis method based on hypergraphs," 12th IFAC Symposium on Fault Detection, Supervision and Safety for Technical Processes Safe Process, Ferrara, Italy, June 2024.

[7] J. Rushby, X. Xu, M. Rangarajan, and T. L. Weaver, "Understanding and evaluating assurance cases," (NASA Technical Report No. NF1676L-22111), 2015.

[8] A. Schwierz and H. Forsberg, "Assurance Case to Structure COTS Hardware Component Assurance for Safety-Critical Avionics," 2018 IEEE/AIAA 37th Digital Avionics Systems Conference (DASC), London, UK, 2018, pp. 1-10, doi: 10.1109/DASC.2018.8569774.

[9] H. Forsberg, A. Schwierz, and K. Lundqvist. "Assurance Strategy for New Computing Platforms in Safety-Critical Avionics," Aerospace Technology Congress 2019, FT2019, 08 Oct 2019, Stockholm, Sweden. 2019.

[10] SAE ARP4754B, "Guidelines for development of civil aircraft and systems," Revised Dec. 2023.

[11] A. Bosio, et al., "Emerging computing devices: Challenges and opportunities for test and reliability," in 2021 IEEE European Test Symposium (ETS). IEEE, 2021.

[12] H. Forsberg, J. Lindén, J. Hjorth, T. Månefjord, and M. Daneshtalab, "Challenges in using neural networks in safety-critical applications," in 2020 AIAA/IEEE 39th Digital Avionics Systems Conference (DASC). IEEE, 2020.

[13] J. A. Bilmes, "Submodularity in machine learning and artificial intelligence," arXiv preprint arXiv:2202.00132, 2022.

[14] C. Schorn, A. Guntoro, and G. Ascheid, "Efficient on-line error detection and mitigation for deep neural network accelerators," In International Conference on Computer Safety, Reliability, and Security, Springer, Cham, 2018, pp. 205-219.

[15] Sparsh Mittal, A survey on modeling and improving reliability of DNN algorithms and accelerators, *Journal of Systems Architecture*, Volume 104, 2020.

[16] B. Lakshminarayanan, A. Pritzel, and C. Blundell., "Simple and scalable predictive uncertainty estimation using deep ensembles," *Advances in neural information processing systems,* 30, 2017.

[17] M. A. Ganaie, M. Hu, A. K. Malik, M. Tanveer, and P. N. Suganthan, Ensemble deep learning: A review. *Engineering Applications of Artificial Intelligence*, 115, 105151, 2022.

[18] A. Mohammed and R. Kora, A comprehensive review on ensemble deep learning: Opportunities and challenges. *Journal of King Saud University-Computer and Information Sciences*, 35(2), 757-774, 2022.

[19] J.M. Cluzeau, X. Henriquel, G. Rebender, and G. Soudain, "Concepts of design assurance for neural networks (CoDANN)," Public Report Extract, EASA AI Task Force and Daedalean AG, Version 1.0, March 31, 2020.

[20] EASA, "Artificial intelligence roadmap – a human-centric approach to AI in aviation," Version 1.0, February 2020.

[21] D. Redman, D. Ward and M. Carrico, "AFE 87 – Machine Learning," Aerospace Vehicles Systems Institute, Final Report, Issue 1.0, May 7, 2020.

[22] AVSI, "Machine Learning Certification," *Aerospace Vehicles Systems Institute*, Project AFE 89.

[23] SAE, "Artificial intelligence in aviation," *SAE Standardization Committee G-34*. [Online]. Available: https://www.sae.org/works/committeeHome.do?comtID=TEAG34 [Accessed: June 26, 2024].

[24] EUROCAE, "Artificial Intelligence," European Organisation for Civil Aviation Electronics, Working Group WG-114.

[25] M. Ammar Ben Khadra, "An introduction to approximate computing," arXiv:1711.06115v2, 2017.

[26] H.B. Barua and K.C. Mondal, Approximate Computing: A Survey of Recent Trends—Bringing Greenness to Computing and Communication. *J. Inst. Eng. India Ser. B 100*, pp. 619–626, 2019.

[27] W. Liu, F. Lombardi and M., Schulte, Approximate Computing: From Circuits to Applications [Scanning the Issue]. *Proceedings of the IEEE*, *108*(12), pp. 2103–2107, 2020.

[28] F. McCardel et al., "Towards a coherent view of evidence in safety assurance," NASA Technical Report, NASA/TM–20230003336, April 2023.

[29] H. Forsberg and A. Schwierz, "Emerging COTS-based computing platforms in avionics need a new assurance concept," Proceedings of 38th Digital Avionics Systems Conference (DASC 2019), San Diego, Ca, USA, September 8-12, 2019.

[30] A. Graß, C. Beecks, S. A. Chala, C. Lange, and S. Decker, "A Knowledge Graph for Query-Induced Analyses of Hierarchically Structured Time Series Information," European Conference on Advances in Databases and Information Systems, Cham: Springer Nature Switzerland, 2023.

[31] FAA, Advisory Circular, "Enhanced flight vision systems," AC No:90-106A, March 2017.

[32] M. Sun, S. Gautham, C. Elks, & C. Fleming, "Characterizing the Identity of Model-based Safety Assessment: A Systematic Analysis," arXiv preprint arXiv:2212.05401, 2022.

[33] Autopilot Review, "Tesla Hardware 3 (Full Self-Driving Computer) Detailed," [Online]. Available: https://www.autopilotreview.com/tesla-custom-ai-chips-hardware-3/ [Accessed June 28, 2024].