Frontiers in Aerospace Engineering

TYPE Original Research
PUBLISHED 14 November 2024
DOI 10.3389/fpace.2024.1475139

Check for updates

OPEN ACCESS

EDITED BY
Zhaodan Kong,
University of California, Davis, United States

REVIEWED BY
Tammer Barkouki,
University of California, Davis, United States
Gang Chen,
South China University of Technology, China

*CORRESPONDENCE
Bastian Luettig,
bastian.luettig@ils.uni-stuttgart.de

RECEIVED 06 August 2024
ACCEPTED 30 September 2024
PUBLISHED 14 November 2024

CITATION
Luettig B, Akhiat Y and Daw Z (2024) ML meets aerospace: challenges of certifying airborne AI. *Front. Aerosp. Eng.* 3:1475139.
doi: 10.3389/fpace.2024.1475139

ML meets aerospace: challenges of certifying airborne AI

Bastian Luettig *, Yassine Akhiat and Zamira Daw

Institute of Aircraft Systems, University of Stuttgart, Stuttgart, Germany

Artificial Intelligence (AI) technologies can potentially revolutionize the aerospace industry with applications such as remote sensing data refinement, autonomous landing, and drone-based agriculture. However, safety concerns have prevented the widespread adoption of AI in commercial aviation. Currently, commercial aircraft do not incorporate AI components, even in entertainment or ground systems. This paper explores the intersection of AI and aerospace, focusing on the challenges of certifying AI for airborne use, which may require a new certification approach. We conducted a comprehensive literature review to identify common AI-enabled aerospace applications, classifying them by the criticality of the application and the complexity of the AI method. An applicability analysis was conducted to assess how existing aerospace standards - for system safety, software, and hardware - apply to machine learning technologies. In addition, we conducted a gap analysis of machine learning development methodologies to meet the stringent aspects of aviation certification. We evaluate current efforts in AI certification by applying the EASA concept paper and Overarching Properties (OPs) to a case study of an automated peripheral detection system (ADIMA). Aerospace applications are expected to use a range of methods tailored to different levels of criticality. Current aerospace standards are not directly applicable due to the manner in which the behavior is specified by the data, the uncertainty of the models, and the limitations of white box verification. From a machine learning perspective, open research questions were identified that address validation of intent and data-driven requirements, sufficiency of verification, uncertainty quantification, generalization, and mitigation of unintended behavior. For the ADIMA system, we demonstrated compliance with EASA development processes and achieved key certification objectives. However, many of the objectives are not applicable due to the human-centric design. OPs helped us to identify and uncover several defeaters in the applied ML technology. The results highlight the need for updated certification standards that take into account the unique nature of AI and its failure types. Furthermore, certification processes need to support the continuous evolution of AI technologies. Key challenges remain in ensuring the safety and reliability of AI systems, which calls for new methodologies in the machine learning community.

KEYWORDS

artificial intelligence, aviation, certification, airborne, assurance, machine learning, overarching properties

1 Introduction

Over the past decade, Artificial Intelligence (AI), especially Machine Learning (ML), has rapidly advanced thanks to improved processing speed, specialized hardware, data availability, and storage. Multiple sectors, like entertainment and retail, have widely adopted AI, for these applications' failures are non-catastrophic, emphasizing performance and usability over safety. In finance, AI operations in stock trading, insurance, and banking face strict timing constraints. While false positives in fraud detection can have serious financial consequences, they do not threaten human life. The safety-critical healthcare domain already employs AI applications. They aid in diagnostic imaging, robotic surgery, clinical decision support, and patient monitoring. These applications are typically less time-sensitive and rely on explainability and human oversight for safety. In the automotive industry, Tesla's release of Fully Self-Driving v12 marks a significant step toward end-to-end AI applications, which are currently in beta testing in the U.S. The automotive industry's safety approach relies on millions of driving hours to demonstrate system safety, with human drivers expected to intervene if problems occur. However, failures regularly occur in self-driving cars: in June 2021 - May 2022 alone, 400 crashes (US Department of Transportation, 2022) occurred in the U.S. with partially automated driver-assist systems. In both automotive and medical devices, human oversight remains crucial for safety.

Despite the complexity of aircrafts as cyber-physical systems, aviation has remained one of the safest transportation modes for decades due to rigorous certification processes supported by exacting standards for aircraft development, testing, operation, maintenance, and inspection. These processes follow best practices, including ARP4754B for system development, ARP4761A for safety assessment, DO-178C for software, and DO-254 for complex hardware components. These standards introduce Design Assurance Levels (DAL) that categorize the criticality of components based on the potential impact of their failure, requiring the applicant to provide more assurance. This process assumes human-designed systems with good intentions. While the aviation industry aims to use AI to enhance operations, current certification standards are not fully applicable to AI technologies. Regulatory bodies are working on guidance for certifying AI-enabled systems, but there is still insufficient practical experience to establish best practices, unlike with software introduction in aviation. Integrating AI into these processes is vital but challenging, as AI requires specific adaptations. Delseny et al. (2021) reviewed the challenges of certifying ML systems, primarily focusing on software aspects. In addition, this paper addresses the aspects of systems, safety, and hardware. Furthermore, current certification efforts were applied to a specific case study.

AI-enabled aviation systems will become viable only when the AI community understands how to achieve assurance through certification and the aviation community comprehends the safety nuances of AI to develop certification standards that provide the necessary system-level assurance. In this context, Machine Learning Operations (MLOps), a set of practices for automating and optimizing the deployment and maintenance of ML models in production, and Explainable AI (XAI), a set of tools that enable users to understand and interpret ML output, provide promising approaches that could bridge the gap between AI and strict certification requirements. MLOps, with its comprehensive capabilities in dataset versioning, experiment tracking, and model registry throughout the entire life cycle - from development and deployment to continuous monitoring — is crucial for ensuring consistency and traceability in certification processes. On the other hand, XAI can enhance the interpretability of AI decisions, offering techniques such as feature importance, local methods, and global methods that could improve the explainability and transparency required for certification (Angelov et al., 2021). XAI techniques are widely used in medical assessment because they are crucial for ensuring that AI-driven medical assessments are not only accurate but also transparent and interpretable by doctors and end users, paving the way for more widespread adoption of AI technologies in healthcare (Loh et al., 2022; Samek et al., 2021).

This paper offers a comprehensive literature review of AI applications in aviation to identify the methods used and their criticality levels. Section three introduces the case study, ADIMA, an AI-enabled system for automated peripheral detection in integrated modular avionics (IMA). Systems with varying criticality may use the ADIMA approach: emergency lighting (DAL A) and cabin lighting (DAL D). Using ADIMA, section three applies two main certification approaches: the W-development process proposed by the European Union Aviation Safety Agency (EASA) and the Overarching Properties supported by the Federal Aviation Administration (FAA). Section four examines the applicability of current certification processes at the safety, system, software, and hardware levels. Finally, we outline open research questions that require attention to enable the development of airworthy AI aviation systems.

2 Materials and methods

2.1 AI-enabled applications in aerospace

In this paper, AI refers to a machine's capability to mimic human intelligence or even surpass it in performing a given task, such as prediction, recommendations, or reasoning, while ML refers to the development of learning algorithms that allow computers to evolve behaviors based on existing data. This section identifies various applications and essential AI technologies in aerospace engineering and analyzes their levels of criticality. To achieve this, a comprehensive and systematic literature review was performed. To select articles for this review, the authors searched Scopus and Google Scholar using the keywords "Artificial Intelligence in Aerospace," "Artificial intelligence," and "Aerospace engineering." The initial search yielded 246 articles. After removing duplicates and filtering for conference papers and peer-reviewed journal articles from 2014 to 2024, 100 articles remained. A thorough full-text screening further narrowed this to 70 articles for detailed analysis. The study reveals several AI applications for aerospace, with the main four being:

1. **Predictive maintenance (34%)**: Uses data from aircraft sensors to predict potential failures, enhancing reliability and safety while reducing maintenance costs and downtime (Stanton et al., 2023; Korvesis, 2017).
2. **Air traffic management (4%)**: Manages aircraft movement in the air and on the ground to ensure safe and efficient traffic flow, involving air traffic controllers, pilots, airports, and various technologies (Liu et al., 2019; Sridhar et al., 2020).

TABLE 1 AI/ML methods used in different aerospace applications.

Area	Paper	Specific application	ML method
Predictive maintenance	Mathew et al. (2017) [1], Jiangyan et al. (2024), Baptista et al. (2021), Kefalas et al. (2021) [2], Boujamza and Elhaq (2022) [3], Vollert and Theissler (2021), Wang et al. (2023) [4], Zhang et al. (2019) [5], Li et al. (2018)	RUL	LSTM, RFE
	Janakiraman and Nielsen (2016) [6], Das et al. (2010) [7], Liu et al. (2023) [8], Zhao et al. (2021a) [9], Lee et al. (2020) [10], Zhong et al. (2021) [11], Jalawkhan and Mustafa (2021) [12], Corrado et al. (2021) [22]	Anomaly detection	NB, LSTM
	da Silva et al. (2022) [14], De Giorgi et al. (2018) [15]	Engine Health monitoring	ANN, SVM
	Ayhan et al. (2013), Que et al. (2019)	Real-time analysis	RNN, LSTM, AutoEncoder
Flight management and operation	Baumann and Klingauf (2020) [16], Huang and Cheng (2022) [17]	Fuel consumption optimization	DT, NN
	Kong et al. (2022) [18], Tang and Lai (2020) [19]	Autonomous landing	CNN, RL
	Woo and Kim (2020) [20], Ouahouah et al. (2022) [21], Zhao et al. (2021b) [22]	Collision avoidance	CNN, RL
	Fala et al. (2023) [23], Chin et al. (2019), Zhang et al. (2021) [24], Leško et al. (2023) [25], Nanyonga et al. (2023) [26]	Flight phase prediction	LSTM
	Chen and Hu (2024) [27], Han et al. (2021) [28], Amit and Mohan (2021) [29], [30], Ducoffe et al. (2023) [31]	Runway identification	CNN
	Nasoulis et al. (2023)	Cabin pressurization system	-
Air traffic management	Lu et al. (2021) [32], Huo et al. (2020) [33], Rebollo and Balakrishnan (2014) [34]	Delay prediction	XGBoost, RF, Gradient Boosting
	Koch et al. (2019); Yang et al. (2019)	Altitude control	LSTM
	Fadlullah et al. (2017), Kato et al. (2019) [35], Corrado et al. (2021) [13]	Air Traffic Control (ATC)	DNN
	Bejarano et al. (2022) [36], Topal et al. (2023) [37], Giovanni et al. (2021) [38]	Human-AI Teaming	CNN, LSTM, ANN
	Ma and Tian (2020) [39], Rohani et al. (2023) [40], Zeng et al. (2020) [41], Shi et al. (2020) [42], Choi et al. (2021) [43], Schimpf et al. (2023) [44],Shi et al. (2018) [45], Jia et al. (2022) [46]	Aircraft trajectory prediction	CNN-LSTM
Security and surveillance	Garcia et al. (2021), Dave et al. (2022)	-	-
Other applications	Gaikwad et al. (2023) [47], Liu and Ferrari (2019) [48]	Autonomous taxiing	CNN, object detection
	Muñoz-Esparza et al. (2020) [49], Chen et al. (2023) [50]	Weather prediction	RF, GBRT
	Bejarano et al. (2022)	Single pilot operation	CNN, LSTM
	Ducoffe et al. (2024), Lazzara et al. (2022),Heidari et al. (2024), Yondo et al. (2019), Sommerwerk et al. (2016), Espinosa Barcenas et al. (2023), Biannic et al. (2016)	DDSM	LSTM, MLP
	Bauranov and Rakas (2021)	UAM	-
	Shakhatreh et al. (2019), Mohsan et al. (2023), Yasin et al. (2020)	UAVs	Deep RL

3. **Flight management and operation (39%)**: Involves planning, executing, and monitoring flight phases, including route planning, fuel management, navigation, and communication with air traffic control. It optimizes flights, enhances safety, and reduces operational costs, contributing to aviation sustainability (Oehling and Barry, 2019).
4. **Security and surveillance (23%)**: Aerospace systems rely heavily on networks and software, making them targets for cyber-attacks and security breaches. Machine Learning enhances security by providing advanced detection, prevention, and response mechanisms (Dave et al., 2022).

Table 1 techniques highlights the application of AI/ML[1] in aerospace (Alzubaidi et al., 2021; Sarker, 2021). Predictive

1 AI-enabled methods used are the following: RFE:Recursive Feature Elimination, NB:Naive Bayes, ANN: Artificial Neural Network, SVM: Support Vector Machine, RNN: Recurrent Neural Network, DT: Decision Tree, XGBoost: eXtreme Gradient Boosting, RL: Reinforcement Learning, DNN: Deep Neural Network, GBRT: Gradient Boosting Decision Tree, MLP: Multi-Layer Perceptron), UAM: Urban Air Mobility, UAV: Unmanned Aerial Vehicle, LR: Linear Regression.

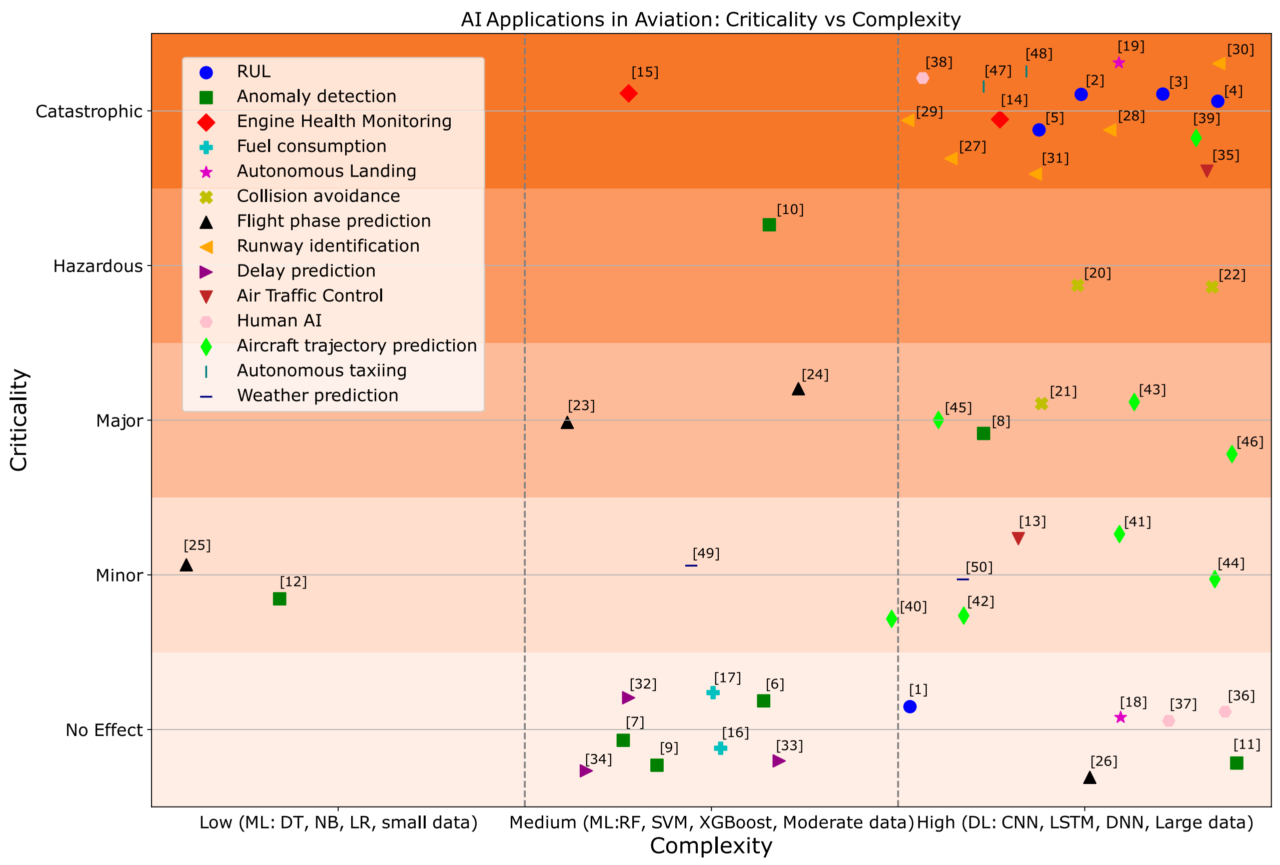


FIGURE 1
AI applications in aerospace: Criticality vs. Complexity, refer to Table 1 to match numbers to references.

maintenance tasks, such as Remaining Useful Life (RUL) prediction and anomaly detection, commonly employ Long Short-Term Memory (LSTM) networks (Lindemann et al., 2021) and Random Forest (RF) (Akhiat et al., 2021). In contrast, flight management and operations—covering areas like aircraft trajectory prediction, autonomous landing, and taxiing—often rely on Convolutional Neural Networks (CNN). Meanwhile, air traffic management tasks like delay prediction and altitude control use methods like XGBoost (Chen and Guestrin, 2016) and LSTM. Researchers choose these techniques based on their suitability for specific tasks and compatibility with the available data. For example, LSTM is favored in predictive maintenance for its ability to capture temporal dependencies in time-series data. At the same time, CNNs are preferred for efficient computer vision processing in applications such as aircraft trajectory prediction and runway detection.

Figure 1 illustrates the relationship between methods' complexity and applications' criticality. The level of assurance evidence required is directly proportional to the application's criticality, and the task's complexity further amplifies the challenge of providing this evidence. The graphic demonstrates a wide range of criticality and complexity, with criticality determined by the impact of potential failures and complexity assessed based on the algorithm and the amount of data involved. From the figure, it's evident that there is a cluster of computer vision applications that are both complex and critical. Among the applications identified in the literature review, only three are classified as hazardous in terms of criticality. For the introduction of AI-enabled applications, those with low criticality and low complexity represent the "low-hanging fruit." However, it's important to consider the business value, as certifying these methods will incur additional costs. Applications with low criticality but high complexity can serve as a testing ground for validating current certification efforts and assurance methods for AI. These efforts can then be applied to more critical applications that might offer a higher business impact.

2.2 Aviation certification process

The certification process follows the Swiss cheese model, which is widely used in aerospace. This model acknowledges system flaws and incorporates multiple defense layers to mitigate risks. These layers, including engineering designs, redundancy systems, maintenance procedures, and training programs, significantly reduce the likelihood of accidents. The system development process begins with the specification of aircraft functions, subsequently allocating these to specific systems and refining them further. At this stage, a comprehensive safety assessment is conducted per ARP4761A guidelines (SAE International, 1996) for civil airborne systems (see Figure 2). This assessment identifies and classifies failure conditions by their impact, ranging from catastrophic to no safety effect. The safety engineer assigns a severity-dependent probability per flight hour for each failure condition the system design must meet. Together, severity and probability define the acceptable risk.

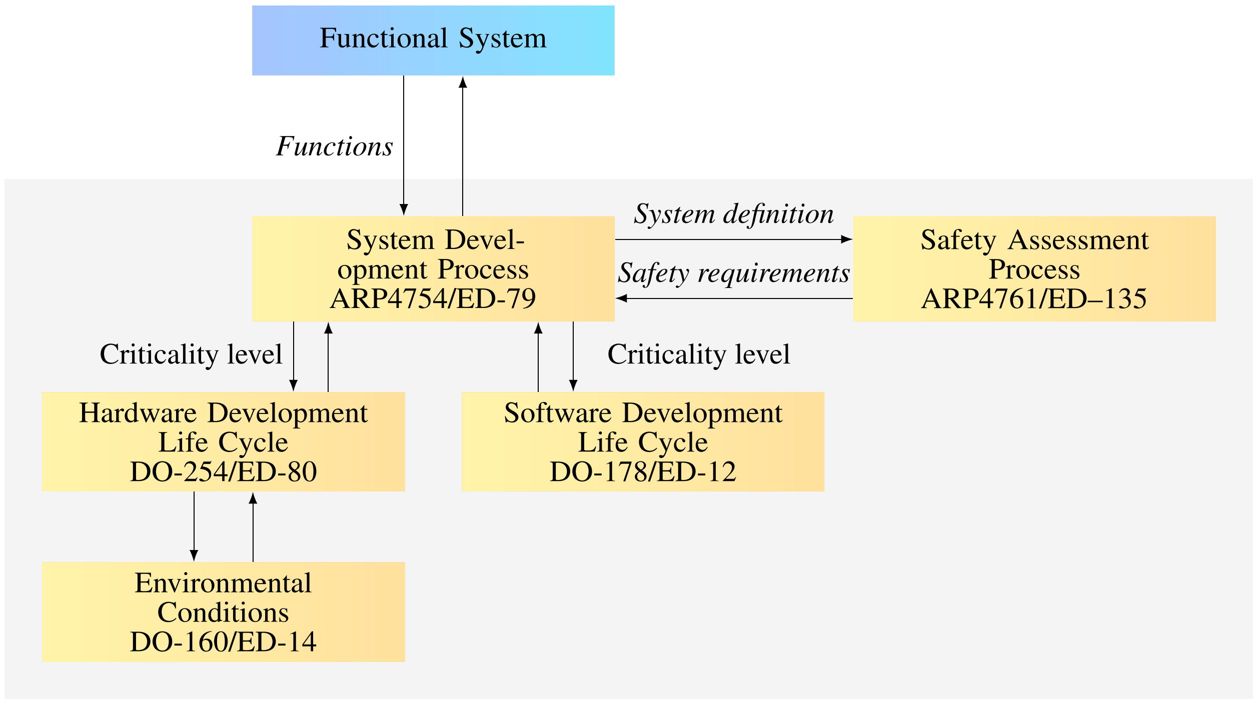

Functional System
Functions
System Devel-opment Process ARP4754/ED-79
System definition
Safety requirements
Safety Assessment Process ARP4761/ED–135
Criticality level
Criticality level
Hardware Development Life Cycle DO-254/ED-80
Software Development Life Cycle DO-178/ED-12
Environmental Conditions DO-160/ED-14

FIGURE 2
Overview of aviation certification process and standards integration.

Building on the safety assessment, safety requirements, criticality levels, and failure information are key inputs for the system development process. ARP4754B (International, 2023) provides comprehensive guidance on system architecture, aircraft-level requirements, design, implementation, verification, and validation, all focusing on safety. This includes incorporating redundancy, Built-In Tests (BIT), monitors, and DAL assignments. The system-level process outputs crucial documents, such as system requirements and architectures.

System and subsystem requirements are allocated to specific software and hardware items. The certification plan at the item level specifies standards for demonstrating compliance, such as DO-178C for software and DO-254 for hardware. Additionally, RTCA has released supplements (DO-331, DO-332, DO-333). While Section 4.1 addresses the application of these standards to AI technologies, new standards or certification processes must align with the existing certification framework.

2.3 Certification efforts of ML applications

This section provides an overview of the initiatives currently being pursued by regulatory agencies. **EASA** has chosen an incremental approach for different autonomy levels with the second version of the concept paper for Level 1 and 2 machine learning applications currently under review (European Union Aviation Safety Agency, 2024). The AI trustworthiness framework comprises AI Assurance, Human Factors for AI, and AI Safety Risk Mitigation. EASA extends the V development process into a W-shape to ensure learning assurance, address data management, model training, verification, and more. The guidance sets high-level objectives for AI-based systems, emphasizing the need to capture specific aspects in specifications for learning assurance. Although the concept paper outlines advanced means of compliance methods, it lacks detailed guidelines for satisfying these objectives. The incremental approach, presented in the EASA roadmaps, allows authorities and applicants to evaluate the standard while applying it to lower criticality and lower autonomy levels and to refine the details based on the gained experience.

The **FAA** employs a bottom-up approach, collaborating closely with applicants to gain practical experience before writing guidelines. Since September 2023, the FAA has organized an AI Roadmap and Technical Exchange Meetings on AI/ML, where they present progress, and experts in ML and aviation share their work and discuss the needs and concerns of AI systems. To support this approach, the Overarching Properties Working Group (OPWG), in collaboration with NASA and the FAA, has developed a framework known as the Overarching Properties (OPs) (NASA/TM–2019–220292, 2019). The OPs consist of three high-level properties designed to provide applicants with a flexible means of proposing novel compliance methods. These properties include: "***Intent***: the defined intended behavior is correct and complete with respect to the desired behavior; ***Correctness***: the implementation is correct with respect to its defined intended behavior, under foreseeable operating conditions; and ***Innocuity***: any part of the implementation that is not required by the defined intended behavior has no unacceptable impact." The OPs framework allows applicants to propose customized compliance methods, enabling authorities and applicants to gain practical experience that informs future standards. Within OPs, arguments are decomposed until reaching a *leaf argument*, where premises are supported by artifacts and require no further decomposition (Daw et al., 2021).

The **EUROCAE/SAE WG-114/G-34** is developing the ARP6983 to guide AI-enabled system development, focusing initially on supervised ML for lower criticality scenarios. It introduces the concept of an ML Constituent (MLC), which includes ML models, traditional software, and hardware items. The group is working on providing guidance for defining the Operational Design Domain (ODD) from the system level through data selection to implementation, integrating ARP with other standards.

2.4 Case study ADIMA

This section presents the case study ADIMA. This case study highlights the main challenges in certifying AI under classical software certification standards. Section 2.4.1 to Section 2.4.4.5 describe the system in detail, covering its architecture, functionality, data, and development process. Due to the space limitations, the requirements, code, data and arguments for the OPs are available in Github[1]

2.4.1 System description and development process

The AI-enabled ADIMA system enables automated peripheral detection based on electrical properties using an autoencoder and a classification network. The autoencoder consists of the encoder and the decoder. The encoder reduces the features to a low-dimensional latent space while the decoder reconstructs the data. In ADIMA, it serves to detect input-anomalies using the mean squared error (MSE) from input to output. On the other hand, the classification network consists of an input layer, multiple hidden layers, and an output layer with a softmax activation function that returns probabilities for the detected classes. ADIMA operates on an IMA platform composed of multiple IMA devices, each equipped with I/O interfaces for connecting peripherals such as LEDs, sensors, motors, or bus systems. Multiple systems share the IMA platform, and wiring issues might occur. Hence, ADIMA must behave robustly when connecting peripherals other than LEDs. An IMA device executes applications inside partitions that provide an ARINC653 compatible API. This system supports highly safety-critical functions in emergency lighting (DAL A) and less critical functions in cabin lighting (DAL D). ADIMA's primary goals are a) to determine if the connected peripheral is among the list of acceptable peripherals and b) to identify a peripheral without adding additional hardware. The LEDs, simple peripherals used as proof of concept, exhibit distinct current-voltage characteristics that vary by color (see Figure 3), allowing for precise peripheral identification. For identifying peripherals, ADIMA employs a data-centric approach by comparing the current-voltage curves of peripherals to reference values obtained from laboratory data. Instead of relying on a single point, ADIMA uses the entire curve for comparison, as this tolerates individual measurement errors.

During platform start, the system scans each IO, computes the MSE, and, if acceptable, classifies the peripheral. It then assigns suitable tasks, and during normal operation, it cyclically executes the assigned tasks, which control the detected peripherals.

2.4.1.1 Operational concept and design framework

The ADIMA concept is defined in Scheme 1.

2.4.1.2 Safety assessment

The safety assessment (as per ARP4761A) requires a functional hazard assessment. System functions:

- Detect peripheral type attached to an IO.
- Detect peripheral degradation attached to an IO.

There are three major failure modes: passive, wrong detection, and out-of-control.

SCHEME 1 Operational concept and design framework.

System intent
Two different applications use ADIMA: • **ADIMA-Cabin Lighting** system (*ADIMA-CL*) ensures the manufacturer installed the correct LED in the aircraft cabin. Non safety Critical. DAL D • **ADIMA-Emergency Lighting** system (*ADIMA-EL*) ensures the manufacturer installed the correct LED in the aircraft's emergency lights. Highly safety critical (CS25.812). DAL A
Operation
ADIMA operates as follows: 1. During production, a worker connects LEDs to the avionics modules. 2. During each bootup, the system assesses the LEDs' remaining lifetime and assigns a degradation class (fully operational, degraded, worn, defective, other). 3. For all worn, defective, and other classified peripherals, the system generates a maintenance report. 4. For all fully operational and degraded LEDs, the system automatically detects the connected LEDs to determine their color. 5. The system assigns the detected LEDs to the relevant functions according to their color and degradation class. 6. The system changes to the normal operation mode and executes the functions
Operation Domain (OD)
The system shall detect four distinct colors for LEDs of one manufacturer (red, green, blue, yellow) with an operational current of 40 mA and a voltage range of 2 V–4 V. The system shall operate within a temperature range of 10°C–60°C in a humidity range of 20%–100%, and at a pressure range of 70 kPa–105 kPa
Operational Design Domain (ODD)
The ML constituent *classifier* accurately detects LED colors for known types with an operational current of up to 40 mA and a voltage range of 2 V–4 V

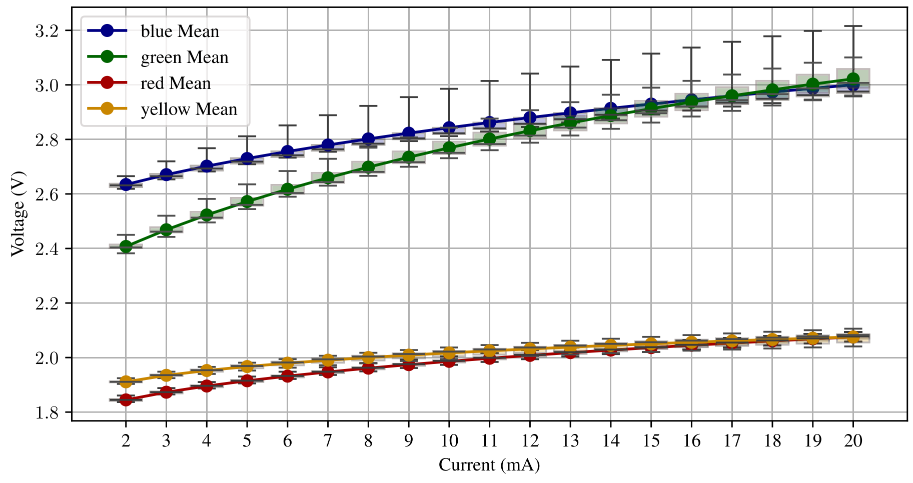


FIGURE 3
Voltages over given currents for the LEDs with error bars. The dots indicate the mean value across all LED measurements; the line shows the interpolated behavior between the measurements. The colored boxes indicate the first and third quartile, and the whiskers identify minimum and maximum values.

1. **Passive Failure:**
   - **Description:** The system fails to detect any peripherals and thus does not execute any desired functions.
   - **Effects:**
     a. **ADIMA-CL:** This failure has no safety effect, as emergency lighting provides sufficient illumination. **(No Safety Effect)**
     b. **ADIMA-EL:** An entire passive system failure is unacceptable (CS.25.812, e.g., due to a common mode error). **(Hazardous)**
   - **Attribution:** Peripheral failures or software errors cause passive failures.
2. **Wrong Detection:**
   - **Description:** The system incorrectly detects peripherals.
   - **Effects:**
     a. Incorrect LED color assignment, causing minor inconvenience. **(No Safety Effect)**
     b. Incorrectly uses a non-LED peripheral to execute a task. Potential fire hazard. **(Catastrophic)**
     c. Incorrectly identifies the health of an LED as broken. Early Maintenance **(Minor)**
     d. Incorrectly identifies the health of an LED as intact. LED is not illuminated, although the system believes it is. **(Major)**
   - **Attribution:** Peripheral failures/changes or software errors cause wrong detection.
3. **Out-of-Control:**
   - **Description:** The system erratically executes tasks on the peripherals that were undefined beforehand.
   - **Effects:** Out-of-limit commands to the peripherals. Potential fire hazard. **(Catastrophic)**
   - **Attribution:** Solely the underlying avionics hardware may cause this failure condition; the relevant hardware safety assessment covers it.

The assessment shows that ADIMA-EL must not fail due to common mode/common cause errors—this requires a dissimilar and redundant approach for individual parts of ADIMA-EL. Wrong classification detection may lead to the system wrongly using peripherals or open IOs for the ADIMA-EL, i.e., the system is not working. Therefore, this failure condition must be addressed with additional architectural means. ADIMA-EL is designed to ensure no single point of failure and to ignore LEDs that are not in good condition while reporting such issues for maintenance.

2.4.1.3 System level requirements and architecture

HR-1: The system shall accurately identify each known LED color and return an error when an unknown peripheral is connected. The system shall predict the health of each LED with minimal 5% false positives and less than 1% false negatives.*Rationale* Incorrect detection may cause hazards: false positives can wrongly mark LEDs as inactive, while false negatives can lead to improper system execution, jeopardizing safety and functionality.

Each ADIMA system consists of a host module for task allocation and multiple proxy modules for IO control (see Figure 4). Each system, ADIMA-CL, and ADIMA-EL, has an individual partition and access rights to specific IOs on each module. Proxy modules detect their peripherals and send this data to the host module, which then allocates tasks accordingly.

ADIMA supports multiple proxy modules in a system (Figure 5): each records the LED curves using the *Recorder* and transmits this data array to the *Autoencoder* to obtain the Mean Squared Error (MSE). If the MSE is acceptable, i.e., the data array is within *IOD* (Inside of Distribution), the *Classifier* uses the data array to compute probabilities for the individual colors, which the *Detector* then converts into an ID. Otherwise, the *Logger* will store a maintenance message for this peripheral. The proxy module then sends all data to the host module, in which the *Allocator* takes the peripherals and allocated suitable tasks to the modules and their peripherals. The *Task Executor* uses the resulting configuration to run the tasks and send commands via the individual *IO Controllers* to the peripherals.

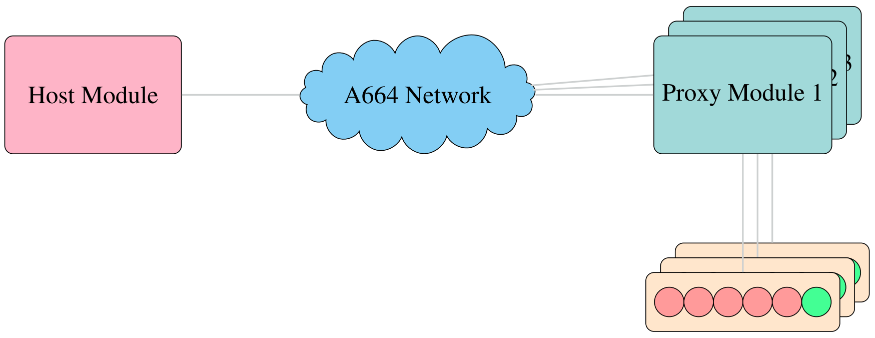


FIGURE 4
Hardware architecture with one host module, the Avionics Data and Communication Network (ADCN), multiple proxy modules with their individual peripherals.

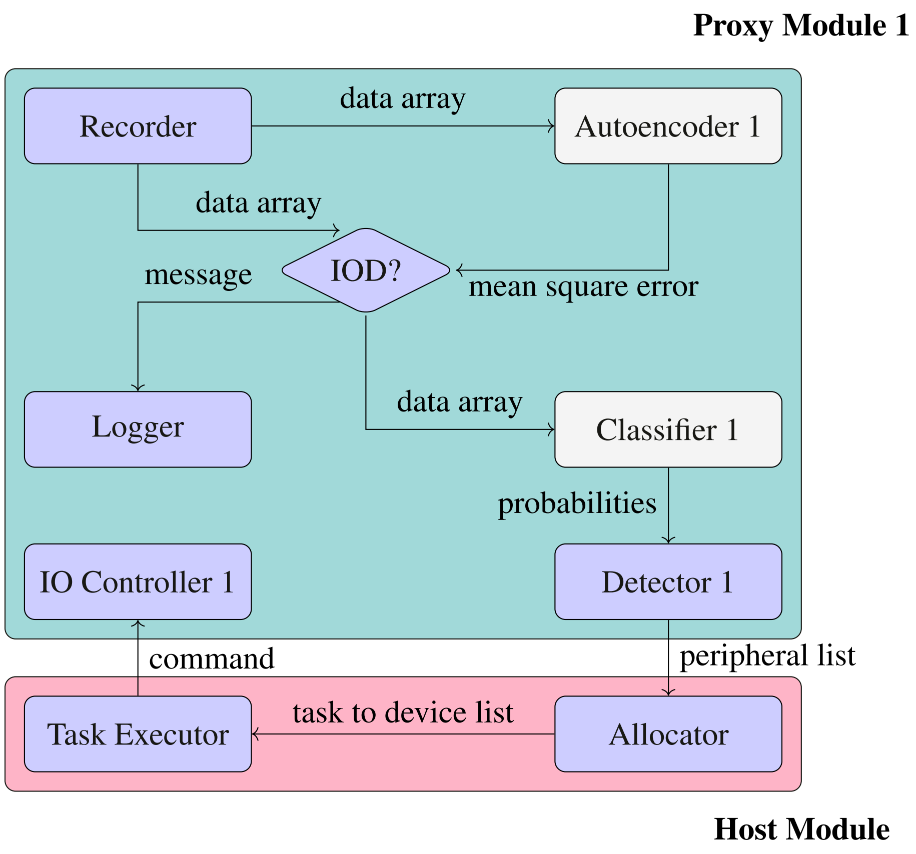


FIGURE 5
ADIMA system architecture with software allocated to the proxy and host module, AI-enabled software in gray, classical software in blue.

This architecture limits the single-point failures to the host module. ADIMA-EL must consist of at least two proxy modules. A safe mode could then simply include the LEDs allocated to the emergency lights to turn on when the host sends no further commands. Multiple host modules can be implemented to enhance reliability, employing a consensus mechanism to ensure a consistent active status (Luithardt, 2017).

The ADIMA system's design reduces common mode errors in proxy modules by leveraging three strategies: module-specific live electrical curve data, approach-specific autoencoders and classifiers for fixed-voltage and fixed-current configurations, and dissimilar hardware modules (types A and B). By utilizing module-specific data, the system holds dynamic dissimilarity: the same error cannot affect all modules due to differences in their inputs. Using unique autoencoders and classifiers for each approach introduces software dissimilarity, as each network is trained on specific data. Additionally, incorporating different types of hardware modules provides hardware dissimilarity: an error in hardware design does not affect all modules simultaneously. Along with the system's capability to track metrics such as network confidence, autoencoder deviation, and input vectors, these arguments contribute to robustness against common mode errors.

2.4.2 Data analysis

2.4.2.1 Training and validation data

A data set is defined by Equation 1 as: a given current c_i results in a voltage x_i stored in $\underline{x}_i$ together with the color label $f(\underline{x}_i)$:

$$\text{Fixed current}: \quad \underline{c}_i = \begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_n \end{pmatrix}_i \rightarrow \begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{pmatrix}_i = \underline{x}_i^{(v)} \tag{1}$$

To account for manufacturing and measurement deviations, the data includes 20 LEDs per color and 20 measurements per LED. Given the equal data distribution across the colors, the dataset is unbiased towards the classification task. Figure 3 shows the standard deviation and the extreme values for each color and current. For a detailed description of data for the fixed-voltage approach, refer to Luettig et al. (2022).

2.4.2.2 Out-of-distribution data (OOD)

OOD refers to the data sampled from a different distribution than one of the training data sets. For ADIMA, the primary challenge in detection was determining whether the captured curves corresponded to the intended devices. For ADIMA-EL, we found that a passively failed detection is acceptable (Github/HR-12) unless this is due to a common mode error (Github/HR-11). The approaches ensemble training, garbage classes, and binary decision networks did not reliably detect out-of-distribution data. On the other hand, Autoencoders used in anomaly detection significantly improved and effectively addressed the OOD challenges in the tested cases (Luettig and Annighoefer, 2023). Training revealed the curves showing only a single distinct feature among the four LED colors, which led to the conclusion that one parameter suffices to describe the $\mathcal{D}^{(c)}$ curves.Training revealed the curves showing only a single distinct feature among the four LED colors, which lead to the conclusion that one parameter suffices to describe the $\mathcal{D}^{(c)}$ curves.

To test the autoencoder's functionality, supplementary testing datasets were generated, including aged LEDs (1,600 rows, 400 per color), motors (400 rows, 20 motors), empty IOs (400 rows, 10% random noise), and physically broken LEDs (4 LEDs, 4 rows). This data was not used for the autoencoder's training or validation.

2.4.2.3 Operational design domain data (ODD)

To ensure adequate coverage of the ODD, we developed a comprehensive list of potential ODD parameters. We evaluated their influence on the LED electrical properties and ADIMA performance as detailed in Table Github/ODD Parameters. For two parameters, cable length and external temperature, we produced additional testing data: one LED per color, covering the temperature range of 30°C to 65°C and an additional cable length range of 0 m–0.8 m.

2.4.3 Model training, learning and implementation

2.4.3.1 ML constituent architecture and training

2.4.3.1.1 Classifier training. The autoencoder and classifier were trained using Tensorflow 2.16.0rc0/Keras 3.0.5 in Python 3.12.4. For the classification network, a hyperparameter study revealed suitable networks small enough to fit within the IMA module's partition (1MByte). The training and validation data were randomly picked from the initial dataset (20 curves x 20 LEDs x 4 colors). The study showed that a single hidden ReLU layer and a softmax output layer provide sufficient accuracy on training and validation datasets. Additionally, ReLU and a limited number of nodes bring the benefit of low computation times and low memory usage. Luettig et al. (2022) details the evaluated architectures, along with their respective validation accuracies and the number of training epochs. Most hyperparameters were fixed as specified in Table 2, while a subset underwent additional hyperparameter optimization.

TABLE 2 Hyperparameters for classification network.

Fixed hyperparameters			
Algorithm	Feed Forward Neural Network (FFNN)	**Epochs**	Until 100% validation accuracy
Hardware	AMD Ryzen 7 PRO 7840U CPU	**Batch Size**	25
Training/Validation Split	80-20, randomized	**Initialization**	He Normal
Loss Function	Categorical Cross-Entropy	**Output Layer**	Softmax
Learning Rate	Default (0.001)	**Dropout Rate**	Not used
Early Stopping	Not used	**Weight Decay**	Not used
Variable Hyperparameters			
Activation Function	e.g., ReLU, Sigmoid, Tanh	**Number of Layers**	e.g., 3, 5, 10
Number of Nodes per Layer	e.g., 4, 10, 19, 99	**Optimizer**	Adam, SGD
Hyperparameters for autoencoder			
Loss Function	Mean Squared Error (MSE)	**Accuracy Function**	Accuracy (MSE)
Input Normalization	Divide by 4		

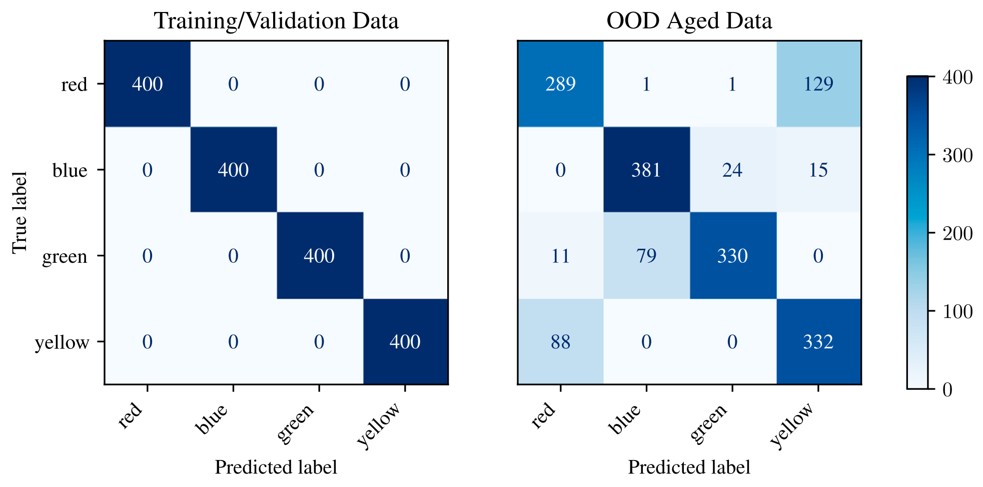


FIGURE 6
Confusion matrices for training/validation data and OOD aged LED data.

2.4.3.1.2 Autoencoder training. A hyperparameter study was conducted for autoencoder training, resulting in selecting a compact network architecture with three hidden layers (5-2-5). This configuration demonstrated favorable accuracy while minimizing computational and memory demands. Autoencoder networks with all sigmoid layers and Adam optimizer performed best in the study. An important task is to normalize the input: for fixed currents: $\underline{x}_i^{(v')} = \underline{x}_i^{(v)}/4$ which yields $\forall i = 0.n$: $0 \leq x_i^{(v')} \leq 1$. Table 2 shows the hyperparameters of the autoencoder.

The training minimizes the MSE, which indicates whether the data falls within or outside the training scenarios. This approach necessitates establishing acceptable residual values. These provide additional information about the extent of training data deviation, indicating the degree of LED degradation. A study on testing data for aged LEDs, failed LEDs, DC motors, and empty IO ports determined that intact LEDs typically exhibit an MSE of $\leq 6 \times 10^{-5}$, aged LEDs that still work of $\leq 1 \times 10^{-4}$, faulty LEDs that do not work of $\leq 1 \times 10^{-3}$, unknown peripherals of $\leq 1 \times 10^{-1}$ and an empty interface of $\geq 1 \times 10^{1}$. Still, these values highly depend on the actual hardware and interfaces used, the actual autoencoder, and its accuracy against the reference data.

2.4.3.2 Implementation

2.4.3.2.1 ML constituent. The implementation must comply with requirements by the module's hardware. The hardware features real-time capabilities and a dated single-core PowerPC CPU. The implementation consists of two components: model-independent functions in traditional software, subject to DO-178C objectives, and the network's parameters, Parameter Data Items (PDI). A script exports these parameters (biases, weights, node counts, and layers) directly from the Tensorflow/Keras model into C structure data (Luettig et al., 2022).

2.4.3.2.2 Implementation of classical software components. The software blocks Recorder, Logger, IO Controller, Detector, Allocator, and Task Executor fall into the traditional software category manually developed in C and thus subject to DO-178C objectives.

2.4.4 Verification

Development activities include verifying the data used for training, validation, and testing, validating the trained model, and finally, verifying the integrated and implemented application. The activities are captured as Test Cases (TC).

2.4.4.1 Training, validation, and testing data

To ensure the data adequately covers the ODD, the data acquisition process incorporates typical engineering variations. This involved collecting data from different LEDs of the same type and making (manufacturing deviation). Additionally, multiple measurements per LED capture measurement deviations.

TC-1 Manually check variance and bias of training and validation data.

2.4.4.2 Trained model

Validation of the autoencoder uses additionally acquired OOD testing data for aged or faulty LEDs, wrong peripherals, and open IOs. The autoencoder identified 14% LEDs labeled *new* as *aged*, which is deemed acceptable, as the manufacturing deviation may also result in a limited lifespan of the LED. Furthermore, the autoencoder identified 16.5% LEDs labeled *aged* as *new*, which is also acceptable, as the LED still functions in that case. The artificial aging process applied out-of-specificationcurrent to the LEDs. For the motors and other clear OOD data, the autoencoder works correctly.

TC-2 Check the confusion matrix of the autoencoder for misidentification on training, validation and testing data.

The classifier, on the other hand, shows perfect scores for the training and validation data; see the confusion matrix in Figure 6 (left). The classifier detects all known peripherals correctly but fails 100% for motors and empty IOs. Moreover, when checking the networks' confidence, it is mostly 100% confident on its wrong classification; see Figure 7 for confidence on correct and misclassifications. When using aged data, the classifier starts to misclassify the LED colors; see Figure 6 (right), which shows the confusion matrix on aged peripherals. The classifier correctly identified 83.3% peripherals. This highlights that the software cannot rely solely on the classifier. Hence, the software needs to

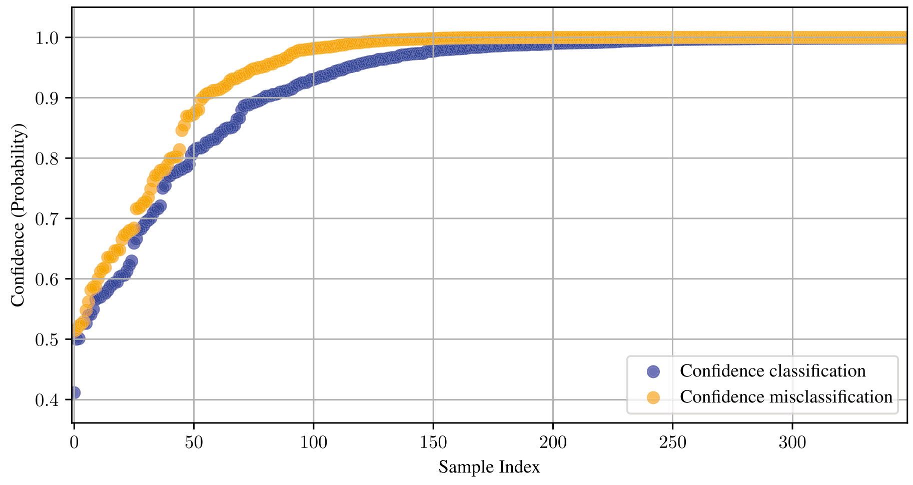


FIGURE 7
Confidence for correctly and incorrectly classified peripherals on OOD Aged Data.

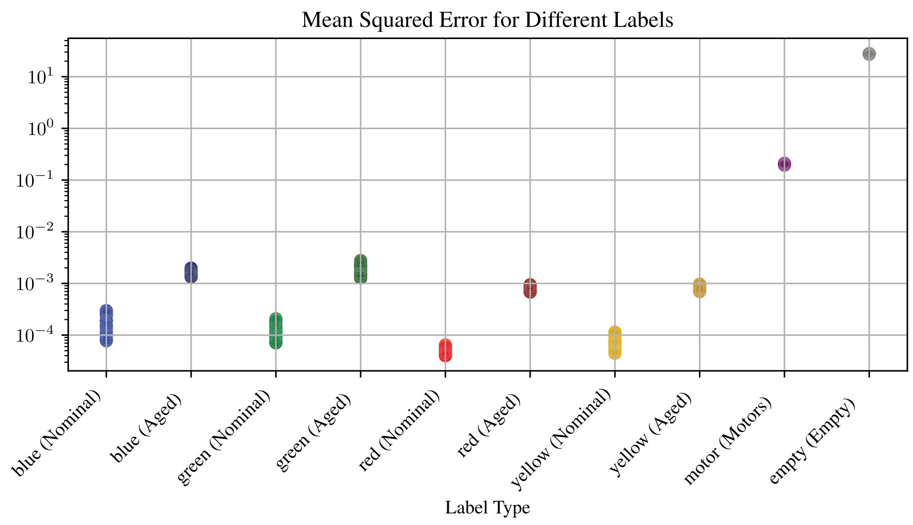


FIGURE 8
Mean Square Error for autoencoder prediction. Light datapoints show training data, and dark datapoints show the results for aged components. The *y*-axis is logarithmic, allowing all results to fit into one diagram.

remove such data early on. Furthermore, Figure 8 shows the MSE for each dataset. Furthermore, it shows that a certain limit can be defined when a component is aged and soon-to-be-broken, a device is unknown, or an IO is open.

TC-3 Check the confusion matrix of the classifier for misclassifications on training and validation data.

TC-4 Check the classifier's confidence for misidentification on training and validation data.

TC-5 Check the confusion matrix of the classifier for misclassifications on OOD data.

2.4.4.3 Implemented application

The export script converts the trained model into C parameter data, extracts each layer definition (activation function, number of nodes, weights, biases), and generates a C source file. The first verification step includes testing on a PC and feeding against training and validation data to expose programming errors. The second verification step tests the actual target hardware with live data, considering the different accuracies of laboratory equipment and IMA hardware.

TC-6 Check the confusion matrix of the implemented application for misclassifications and misidentifications on training, validation, and testing data.

2.4.4.4 ADIMA system

The final verification step involves the actual integrated system that consists of:

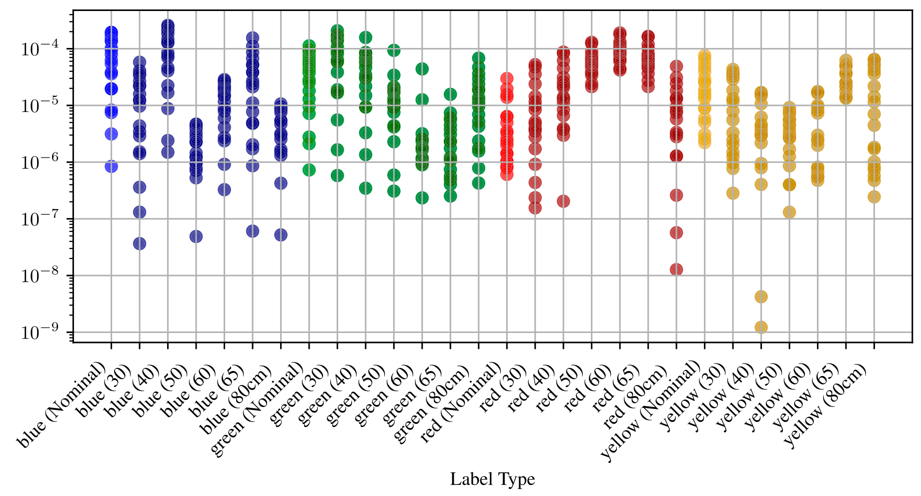
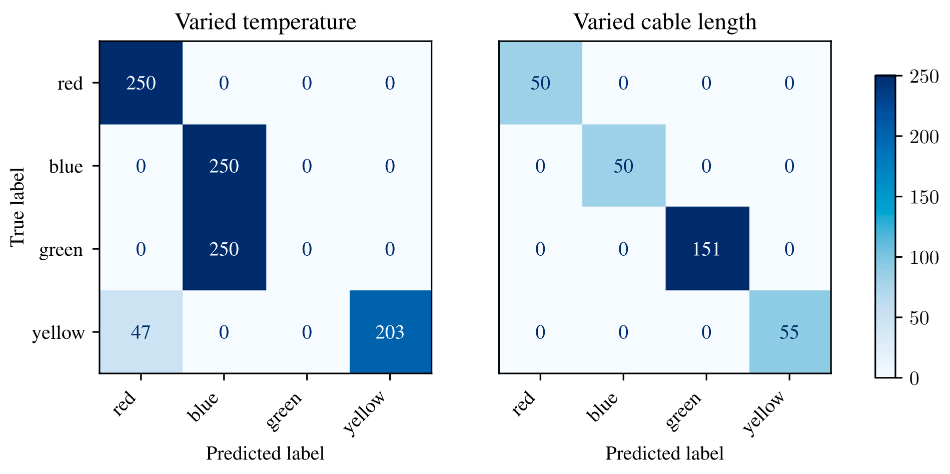


FIGURE 9
Mean Square Error for autoencoder prediction on varied cable and temperatures, the *y*-axis is logarithmic, it shows the training data (light color) and the environmentally altered cases (dark color).



Varied temperature	red	blue	green	yellow
red	250	0	0	0
blue	0	250	0	0
green	0	250	0	0
yellow	47	0	0	203

Varied cable length	red	blue	green	yellow
red	50	0	0	0
blue	0	50	0	0
green	0	0	151	0
yellow	0	0	0	55

FIGURE 10
Confusion matrices for ODD robustness data.

1. 1 host partition on a VCE type B
2. 1 proxy partition on the VCE type B
3. 1 proxy partition on the VCE type A
4. 3 LEDs on VCE type A, 4 LEDs on VCE type B

The host partition executes the *Allocator* and *Task Executor* shown in Figure 5, the two proxy partitions execute the respective autoencoder and classifier, and the system needs to correctly identify the possible tasks using the identified peripherals. There are three possible tasks to allocate: *Task 1* requires two red LEDs, *Task 2* requires one red, one green, one blue LED, and *Task 3* requires two blue LEDs. *Task 2* is the most important, then follows *Task 1* and *Task 3*. The tasks send distinct commands to the LEDs so the observer can tell them apart. All tests passed.

2.4.4.5 ODD robustness

Additional testing data was acquired to verify robustness with ODD data for ADIMA-EL and ADIMA-CL. The data included both, cable length (0.2 m.1 m) and environmental temperature (20°*C*-65°*C*). Figure 9 shows an excerpt of the autoencoder output for additional data. Evidently, the MSE values remain in the same order of magnitude and thus prove no significant influence of these variables on the autoencoder.

The classification network processes the additional testing data (see confusion matrix in Figure 10). The findings indicated a different type of green LED in the temperature measurement data, which exhibits behavior similar to that of a blue LED. The autoencoder did not catch the anomaly because the LED behaves like a regular blue LED. For the cable length measurement, the data includes an additional three nominal green LEDs, hence the surmount of data for this label. For the temperature dependent measurement, the yellow LEDs show some misclassification on higher temperatures. This requires further inspection.

3 Results

This section demonstrates the compliance matrix of the ADIMA development process for the EASA concept paper's objectives and a structured argument demonstrating how ADIMA holds the OPs.

3.1 EASA W-Process

This subsection presents a potential approach to how to fulfill the latest guidelines from EASAEuropean Union Aviation Safety Agency (EASA) (2024) with the help of the previously stated activities during development. The section shows gaps and provides specific assumptions. EASA has extended the W-process outlined within CoDANN I and II (EASA and Daedalean, 2020; EASA and Daedalean, 2024) with objectives to fulfill; see Section 2.3. The approach resembles the RTCA DO-178C approach for software development, i.e., design assurance. The objectives include well-known good practices from ML and the safety-critical avionics domain. This section will provide an excerpt on the means of compliance. There is no specific ML hardware in the subsystem. The developer must classify the AI-based subsystem as per **CL-01**.

CL-01: We consider the used AI as Level 3B because the system is hidden from all potential crew; its impact is limited.

There are objectives already covered by specific sections that the authors deem not applicable, so an objective may appear in the compliance list and in the not applicable list. For three objectives, multiple sections or test cases help in fulfilling these. We deem Human-Factors requirements (HF) not applicable, as this system does not actually interact with the Flight Crew. We apply the same reasoning to the Explainability requirements (EXP), as ADIMA does not target interactive explainability. Additionally, Ethics requirements (ET) do not apply as there is no interaction with the Flight Crew.

Table 3 provides an overview on how the objectives can be covered by the means shown in this article. Not covered: CO-02, SA-01, DA-03, DA-07.10, DM-08, LM-03, LM-04, LM-11, LM-14, LM-16, IMP-02.06, IMP-12, QA-01.

Not applicable: CO-05, IS-03, ET-*, RU-01, RU-02, RU-03, SU-01, SU-02, EXP-*, HF-*

From the list of uncovered objectives, some do not apply to a DAL D system (ADIMA-CL): DM-08, LM-03, LM-04, LM-11, LM-14, LM-16, IMP-12; others directly address writing plans, validation or verification procedures, which lie outside of the project scope. Select objectives require independence for higher DAL systems (A&B); given the author's organization, this could not be satisfied. The process covers *independence* to some level by shifting the analysis to other personnel: person 1 performed the training/validation data acquisition, person 2 acquired the testing and out-of-distribution data, and person 3 acquired the ODD robustness data. Person 2 performed analysis and implementation.

Working with the new EASA AI assurance guidelines proved to be quite similar to working with the DO-178C and general software development in the safety-critical domain. Due to the ADIMA systems being fully autonomous, the guidelines are not yet fully developed, and the community will find additional objectives in version 3. The current objectives reflect good practice in working with machine learning applications. The overall assessment, just as the DO-178C approach, is based on engineering judgment (safety assessment) and experience. Currently, the development process must fulfill most objectives for any software DAL. The guidelines focus on explainability and human factors - specific to pilot support systems and trace to any AI applications in the medial field (Food and Drug Administration, 2021). In summary, we believe that the guidelines provide a valid step towards integrating AI applications in aircraft in the sense of applications that directly support the pilot in flying while simultaneously providing reasoning and explanation why the AI-based system makes a specific suggestion. However, we cannot fully apply this to the case study in Section 2.4.

3.2 OPRA

This subsection presents an OP-related argument (OPRA) for the Machine Learning Component (MLC)-ADIMA system used in ADIMA-EL, which has been assigned DAL A. The argument structure is shown in Figure 11. Blue propositions represent those addressed during the development process. Red dotted propositions indicate those that the development process did not address but are necessary to support OPs; thus, they pose as defeaters in the argument. The OP-related argument is shown in Scheme 2. Finally, this section provides a table of evidence that supports the leaves of the argument.

MLC-ADIMA refers to the machine learning component within the ADIMA System. While the OPRA provides compliance methods for MLC-ADIMA, the rest of the system follows the existing standards. **Intent** decomposes into high-level and low-level categories. High-level Intent, reflecting high-level requirements, specifies what the MLC-ADIMA needs to achieve by detailing the required performance criteria, the ODD where these criteria must be met, and the necessary data distributions. At the low-level Intent, we refine the high-level intent by outlining non-functional requirements such as memory and time allocation, model requirements, generalization requirements, and data management. Model requirements define properties of the training process, including hyperparameters, training procedures, training data, and initialization. Generalization requirements specify how to handle relevant data not seen during training, both within and outside the ODD, using an autoencoder for monitoring in and out-of-distribution data.

Correctness proposes a requirement-driven testing approach, addressing the requirements defined in the Intent. It is important to note that the test must clearly define testing data. The test cases specify data distributions and evaluation metrics to support this approach. All testing activities must be performed on the target hardware using real data, not simulations. In MLC-ADIMA, unintended behaviors resulting from the use of neural networks are identified as issues related to stability,

TABLE 3 Compliance matrix for ADIMA development against EASA AI assurance objectives.

Section	Objective																													
	CO – 03	CO – 04	CO – 06	CL – 01	SA – 02	SA – 03	IS – 01	IS – 02	ET – 04	ET – 06	DA – 01	DA – 02	DA – 04	DA – 05	DA – 06	DM – 01	DM – 02 – SL	DM – 02 – UL	DM – 03	DM – 04	DM – 05	DM – 06	DM – 07	LM – 01.08	LM – 01.08	LM – 09, 10, 12, 13	IMP – 01	IMP – 07	IMP – 08	IMP – 09	IMP – 10	IMP – 11	CM – 01	EXP – 04.09
2.4.1	x																																	
2.4.1		x	x																															
2.4.1.2					x	x	x	x																										
2.4.1.3															x																			
2.4.1.3												x	x	x		x	x	x	x															
2.4.2																						x												
2.4.3											x						x	x	x	x	x			x		x								x
2.4.3.2																											x						x	
2.4.4				x						x													x											
2.4.4.2																									x									
2.4.4.4																																x		
2.4.4.5																															x			
TC-1									x																									
TC-6										x																		x	x	x				

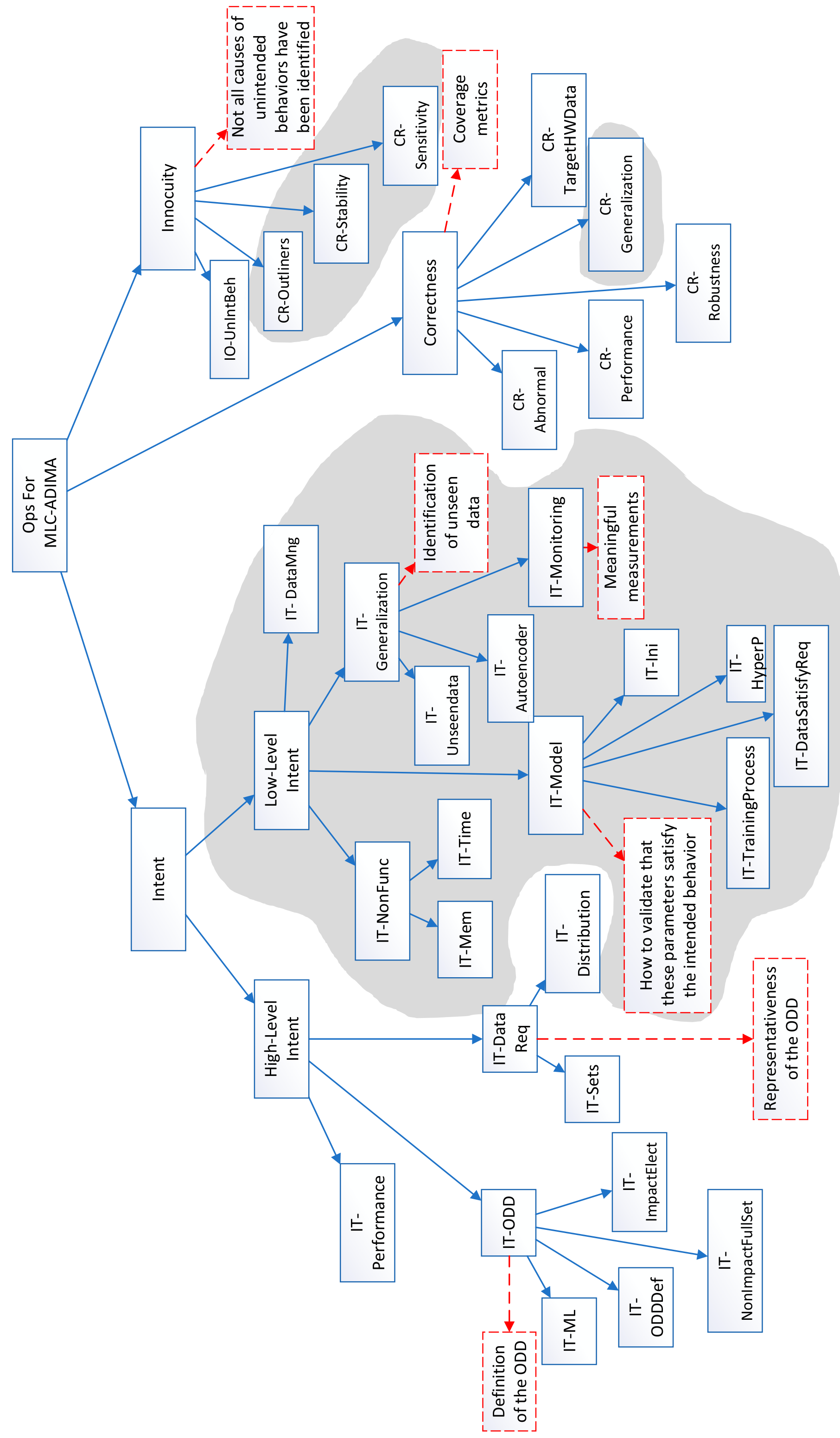


FIGURE 11
OPRA for ADIMA-EL with defeaters highlighted in red dotted lines. Grey part is not necessary for ADIMA-CL.

generalization, out-of-distribution data, data drift, and the incompleteness of data or ODD. While the majority of these issues are addressed in the Intent, additional activities have been added to manage the remaining concerns. These activities include demonstrating the model's stability in the presence of noise, understanding the model through sensitivity analysis using explainability methods, and analyzing all outliers to provide thorough reasoning.

SCHEME 2 OPs argument for ADIMA.

Bindings

Desired behavior (DeB): The ADIMA system shall identify devices, i.e., the color of an LED using its electrical properties during runtime with an accuracy of 100% (-1e-6). It must ensure reliability by eliminating any single point of failure. Additionally, the system shall ignore LEDs that are not in good condition and generate a report detailing the issue

Intended behavior (DIBa): The intended behaviors for DAL A are recorded by relevant data, the required performance, memory requirements, and generalization requirements. This is grouped as high-level intent plus low-level intent.

Intended behavior (DIBd): The intended behaviors for DAL D are recorded by data distribution and the required performance. This is grouped as high-level intent. Device: Any peripheral connected to the IMA hardware module

Data: Refers to the input information used to train, test, and validate models. It consists of arrays for current and voltage that represent electrical phenomena and a label that indicates the color and health status of the LED. The current vs. voltage curve is specific to each type of LED color, varies with age, and changes with the health status of the LED

ODD: The Operational Design Domain defines the specific conditions and environments where ADIMA is designed to operate as intended. Reference to table. Representativeness of the data refers to how accurately a dataset reflects the characteristics and diversity of the entire ODD

Foreseeable operational conditions (FOC): are defined by the ODD, and additional possible conditions are not represented in the model development.

Implementation: Implementation is the integration of the entire ADIMA application (classification network, autoencoder, and integration software) within the ARINC653 partition on the actual IMA hardware module. All runtime data for this implementation comes from the actual IMA module interfaces

Nominal conditions: LEDs from a given supplier defined in the ODD Off-nominal conditions: LEDs from a given supplier are defined in the ODD but with altered characteristics, including aging or cable breaks

Outside of ODD conditions: Any devices not specified in ODD, e.g., motors, LEDs from other suppliers, or other LED colors

Intent

Believing

MLC-ADIMA holds the Intent OP{Intent}

Is justified by applying

MLC-ADIMA defined high level intended behavior is correct and complete with respect to the DeB

To these premises

MLC-ADIMA defined high level intended behavior is correct and complete with respect to the DeB {IT-HighIntent}

MLC-ADIMA defined low level intended behavior is correct and complete with the high-level intent {IT-LowIntent}

Believing

MLC-ADIMA defined high level intended behavior is correct and complete with respect to the DeB

To these premises

The ODD is correct and complete with respect of the DeB {IT-ODD}

Data requirements satisfy the ODD {IT-Data}.

MLC-ADIMA has an accuracy for 100% for data within the ODD {IT-Performance}

Believing

The ODD is correct and complete with respect of the DeB {IT-ODD}

To these premises

List of factors that can affect electrical phenomena {IT-ImpactElect}

List of factors that can affect machine learning model performance {IT-ML}

List of factors that do not impact the MLC-ADIMA estimation {IT-NonImpactFullSet}

Definition of the ODD parameters (including ranges and distributions) for the subset of factors that do affect performance {IT-ODDDef}

(Continued in next column)

SCHEME 2 (*Continued*) OPs argument for ADIMA.

Intent

Believing

Data is representative of the ODD {IT-Data}

To these premises

The data set is randomly divided into training (80%) and cross validation (20%) {IT-Sets}

Training set and cross validation sets show the same data distribution of the ODD {IT-Distribution}

20 measurements are acquired for the same device {IT-Measurement}

Believing

MLC-ADIMA defined low level intended behavior is correct and complete with the high-level intent {IT-HighIntent}

To these premises

Non-functional requirements are correct and complete {IT-NonFunc}

Model requirements are correct and complete {IT-Model}

Generalization requirements are correct and complete {IT-Generalization}

Intent-Cont

Believing

Non-functional requirements are correct and complete {IT-NonFunc}

To these premises

Execution time requirements are correct and complete {IT-Time}

Memory requirements are correct and complete {IT-Mem}

Believing

Model requirements are correct and complete {IT-Model}

To these premises

Model initialization has been captured {IT-Ini}

Hyper-parameters provide a good balance between variance and bias {IT-HyperP}.

Risk of the training processing have been mitigated {IT-TrainingProcess}

Data satisfies intended distribution {IT-DataSatosfyReq}

Believing

Generalization requirements are correct and complete {IT-Generalization}

Is justified by applying

Any part of the executable MLC-ADIMA that is not required by the DIB has no unacceptable impact

To these premises

Autoencoder within the MLC-ADIMA systems identifies off-nominal devices with a false negative rate of x {IT-Autoencoder }

Monitoring requirements for identify changes within the ODD and the data distribution {IT-Monitoring}

Relevant unseen data has been identified {IT-Unseedata}

Correctness

Believing

MLC-ADIMA holds the Correctness OP{Correctness}

Is justified by applying

MLC-ADIMA implementation is correct w.r.t DIB, under FOC

To these premises

MLC-ADIMA implementation satisfies performance requirements within Nominal conditions {CR-Performance}

MLC-ADIMA implementation identifies off-nominal conditions and reports them {CR-Abnormal}

(Continued on following page)

SCHEME 2 (*Continued*) OPs argument for ADIMA.

Correctness
MLC-ADIMA implementation is robust for outside of ODD conditions {CR-Robustness}
MLC-ADIMA implementation can generalize with DeB {CR-Generalization}
Target hardware and real data are used for correctness activities {CR-TargetHWData}
Innocuity
Believing
MLC-ADIMA holds the Innocuity OP{Innocuity}
Is justified by applying
Any part of the executable MLC-ADIMA that is not required by the DIB has no unacceptable impact.
To these premises
Unintended behaviors caused by the implementation choice of using neuronal networks are identified as stability, generalization, outside of distribution, data-drift, incompleteness of the data or ODD {IO-UnIntBeh}.
MLC-ADIMA performance does not get affected by adding 10% of noise to Data {IO-Stability}
MLC-ADIMA performance does not get affected by changes in the network {IO-Sensitivity}
Outliers do not affect the safety of the system {IO-Outliers}

Defeater: "Definition of the ODD" refers to the challenge of validating that the ODD is complete. Developers conducted an exhaustive analysis to understand the factors affecting the current/voltage curve, thereby specifying the ODD and the data distribution required for training, as highlighted in the defeater "Representativeness of the ODD." Additionally, defining corner and edge cases is challenging across multiple dimensions, and finding data for these cases is difficult. In the IT-model conclusion, the goal was to capture all parameters that could affect the model's behavior, enabling reviewers to assess the adequacy of these parameters during validation. However, a defeater arises since it remains unclear which parameters and analysis methods are sufficient for effective validation. It is necessary to identify devices outside the ODD. Developers must first identify potential types of devices or situations (e.g., aging) to provide evidence that the system fulfills this requirement. There are no guidelines for this process, making it difficult to verify if developers missed any possibilities. Although IT-Monitoring attempts to address this issue, extending the ODD, particularly to include non-LED devices, is not achievable with measurements alone; additional information is required. These challenges emerged as two defeaters in the Low-level intent. The defeater "Coverage metrics" refers to the lack of quantitative metrics for the sufficiency of testing data. In **Innocuity**, the aim is to capture all possible causes of unintended behaviors and mitigate them either during development or through analysis. However, developers are uncertain whether all potential sources of unintended behaviors have been covered.

The leaf conclusions are supported by multiple pieces of evidence, with the quantity and depth of evidence varying according to the DAL. Each piece of evidence is linked to its corresponding certification plan. Table 4 demonstrates how ADIMA results map to these leaf conclusions. The ADIMA repository contains the complete list of evidence supporting each argument.

4 Discussion

This section discusses the findings on certification with respect to the literature research and the case study. Section 4.1 outlines the current certification practise and how this relates to AI-enabled systems, Section 4.2 discusses the findings during application of W-process and OPs towards the case-study and Section 4.3 provides an overview on the upcoming challenges that the community still needs to address.

4.1 Applicability of aviation certification processes to AI systems

While AI is merely an implementation choice within aviation systems, the current certification processes for AI might not be directly applicable. This section aims to analyze the applicability of current certifications standards to AI-enabled systems.

4.1.1 Safety considerations

Since ML is a software implementation choice, the initial assumption might be that it only impacts software aspects. However, unlike classical software, ML applications exhibit probabilistic behaviors. ARP4761A (SAE International, 1996) defines an error as "an omitted or incorrect action by a manufacturer, crew member, or maintenance person, or a mistake in requirements, design, or implementation." Such errors can lead to failures during the operation of the aircraft or system, preventing it from behaving as intended. A requirement might specify that an ML model needs to recognize an object with a 98% probability. During the verification process, testing results can demonstrate that the model satisfies this requirement based on the validation data set. Additional metrics, such as precision and recall[2], provide a deeper understanding of the model's false positive and negative behaviors. From a safety perspective, it is crucial to ensure that the remaining 2% probability does not compromise system safety. This assurance must also extend to unseen data (data not included in the validation set). Accurately predicting the performance of an ML model on unseen data remains an active area of research. Requirements also define the operational design domain, specifying where the system must meet performance criteria. However, ML performance is not strictly bounded by the operational design domain but rather by the data distribution for which performance metrics are guaranteed. This distinction becomes apparent in challenges such as data or concept drift. During operation, data drift can cause the ML component to deviate from its intended behavior, leading to failure. This can occur even if the developer initially implemented and designed the system perfectly, as environmental conditions may change. Like hardware failures, this is a stochastic process occurring over time,

2 github.com/ils-stuttgart/adima

TABLE 4 Traceability for OP leafs to development and assessment.

Leaf argument	Description	Section
IT-ML	List of factors that can affect machine learning model performance	2.4.1
IT-ODDef	Definition of the ODD parameters (including ranges and distributions) for the subset of factors that do affect performance	2.4.1
IT-NonImpact FullSet	List of factors that do not impact the MLC-ADIMA estimation	2.4.2, 2.4.4.5
IT-ImpactElect	List of factors that can affect electrical phenomena	2.4.2, 2.4.4.5
IT-Sets	The data set is randomly divided into training (80%) and cross validation (20%)	2.4.3
IT-Distribution	Training set and cross validation sets show the same data distribution of the ODD	2.4.1
IT-Measurement	20 measurements are acquired for the same device	2.4.2
IT-Mem	Memory requirements are correct and complete	2.4.3.2
IT-Time	Execution time requirements are correct and complete	2.4.3.2
IT-Training Process	Risk of the training processing have been mitigated	2.4.3
IT-Data SatisfyReq	Data satisfies intended distribution	2.4.1
IT-HyperP	Hyper-parameters provide a good balance between variance and bias	2.4.3
IT-Ini	Model initialization has been captured	2.4.3
IT-Unseendata	Relevant unseen data has been identified	2.4.2
IT-Autoencoder	Autoencoder within the MLC-ADIMA systems identifies off-nominal devices with a false negative rate of x	2.4.4.2
IT-Monitoring	Monitoring requirements for identify changes within the ODD and the data distribution	–
CR-Abnormal	MLC-ADIMA implementation identifies and reports off-nominal conditions	2.4.1, 2.4.1.3
CR-Performance	MLC-ADIMA implementation satisfies performance requirements within Nominal conditions	2.4.4.2
CR-Robustness	MLC-ADIMA implementation is robust for outside of ODD conditions	2.4.4.5
CR-Generalization	MLC-ADIMA implementation can generalize with DeB	2.4.4.2
CR-TargetHWData	Target hardware and real data are used for correctness activities	2.4.4.4
IO-UnIntBeh	Unintended behaviors caused by the implementation choice of using neuronal networks are identified as stability, generalization, outside of distribution, data-drift, incompleteness of the data or ODD	2.4.4.2
CR-Outliers	Outliers do not affect the safety of the system	2.4.4.2
CR-Stability	MLC-ADIMA performance does not get affected by adding 10% of noise to Data	
CR-Sensitivity	MLC-ADIMA performance does not get affected by changes in the network	

but unlike hardware failures, it is not only related to aging. The safety impact of deficient performance on unseen data can be mitigated either by implementing a runtime monitor or by ensuring a very low probability of such situations occurring. From a certification perspective, two options are available:

1. Connect safety assessment and system requirements through the requirement process and design assurance.
2. In addition to the first option, integrate the reliability values of an ML-enabled system into the safety assessment process, such as through Failure Tree Analysis (FTA).

Option 1 requires no changes to the current safety standards. However, option 2 would necessitate extending the standards to include methods for calculating and integrating reliability values into the safety assessment process. Currently, reliability is determined using operational data, distribution functions, and Monte Carlo simulations to calculate the probability per flight hours. For ML methods, calculating reliability remains an open research question.

Redundancy and monitoring are common architectural mitigations that allow combining lower DAL subsystems to achieve a higher DAL for the overall system (a.k.a. DALgebra). However, standards need to evaluate whether this approach is applicable to ML-enabled systems, considering the lack of proof that independence can be maintained and whether non-performing components can be reliably identified.

4.1.2 System considerations

The next challenge is to address the system itself, ensuring that mitigation mechanisms and criticality allocation remain effective for ML-enabled systems. Aircraft systems employ redundancy to achieve high availability, allowing for dispatchability with malfunctioning components. To ensure that redundant components do not fail simultaneously, they must adhere to the independence principle, which mandates that these components do not share common points of failure. For software-based components, dissimilarity reduces common points of failure by utilizing different processing units, compilers, programming languages, or development methodologies. This approach decreases the probability of an undetected or dangerous system failure due to common software bugs. In the context of ML, this raises the question of how to effectively ensure model dissimilarity in all aspects of ML systems, including training data, hyperparameters, model topology, and initialization.

Runtime assurance can ensure safety by switching to a more straightforward, safer system if a malfunction is detected. However, implementing such monitoring in ML systems is challenging due to the complexity of mapping inputs to outputs, making it difficult to develop dissimilar backup systems or estimate performance reduction. ML algorithms can provide confidence values, but these can be unreliable for unseen data (Hendrycks et al., 2021). Alternative approaches, such as detecting ODD data using distance measures, autoencoders, or trained networks, are under active research.

4.1.3 Software considerations

Software regulations, defined in DO-178C, rely upon the foundational principles of specification, division of concerns, traceability, compliance, and verification. ***Specification*** in software development consists of three levels. System-level requirements outline the overall intent of the software. High-level requirements (HLR) combine this intent with architectural considerations, addressing safety implications. Low-level requirements focus on specific design details, guiding the actual coding and integration of system components. This enables the ***division of concerns***, thereby allowing developers to effectively address concerns at each stage of development and validation. Furthermore, the developer must provide DAL-dependent assurance for each level of requirements. For instance, DAL D may only require assurance at the HLR stage, whereas from DAL C onwards, the objectives cover assurance at all levels. In the context of ML, behavior specifications are not directly outlined; instead, they are defined indirectly through datasets, metrics, and algorithms. Therefore, the community needs to extend standards to address how this intermediate specification of behaviors is managed, subsequently impacting the required level of assurance.

Traceability is the ability to link various artifacts throughout the software lifecycle, encompassing requirements, design, implementation, testing, and maintenance, thereby enabling quality assurance and compliance verification. In traditional software development, traceability connects requirements directly to system behavior, ensuring that each requirement is addressed in the final product. Once traceability is established, validation activities ensure that lower-level artifacts (e.g., test cases) comply with the higher-level artifacts from which they were derived. For ML systems, while we can define requirements, traceability extends from high-level requirements to data requirements and performance requirements. The actual data then serves as the low-level requirement layer. This involves refining these requirements into specific datasets, statistical analyses, hyperparameters, and validation metrics. Despite this, there remains a gap between the specified requirements and the actual behavior of the ML system, highlighting the complexity and challenges in ensuring traceability within ML projects. Furthermore, in the context of ML, this involves methods that validate datasets to ensure they meet specified requirements and that the testing datasets effectively exercise and fulfill those requirements.

Verification, as focused on testing in DO-178C, is approached from two primary perspectives: top-down and bottom-up. The top-down approach emphasizes requirement-based testing, ensuring that all specified requirements are met through tests that exercise the requirements using nominal inputs and assess component robustness under off-nominal inputs. In the context of verifying ML systems, properties of nominal and off-nominal test datasets (e.g., distribution) need to be defined. Due to the nature of ML, the compliance of properties such as stability and generalization, in addition to robustness, must be validated. The bottom-up approach analyzes the implementation structure to detect unintended behaviors, incomplete requirements, or missing tests. Implementation coverage is assessed through structural coverage metrics such as statement coverage or modified condition/decision coverage (MC/DC). In software standards, these metrics provide a means to measure the sufficiency of testing, which depends on the DAL. In the context of ML, determining metrics for the sufficiency of testing remains an open question. Similarly, for coverage metrics, studies have shown that using neurons as analogies for lines of code does not provide the same coverage of the behavior.

4.1.4 Hardware considerations

In hardware development, a critical consideration is whether the hardware item is simple or complex. Hardware incorporating microprocessors or FPGAs is generally classified as complex, necessitating design assurance in compliance with RTCA DO-254. However, a hardware item may be deemed simple, allowing the developer to bypass design assurance requirements, even for ASICs or PLDs. To justify this classification, it must be demonstrated that *a comprehensive combination of deterministic tests and analyses appropriate to the design assurance level ensures correct functional performance under all foreseeable operating conditions with no anomalous behavior* (RTCA, Inc, 2000). For most AI-accelerating devices, it is essential to assume their complexity and develop them in accordance with Design Assurance per DO-254. A recent study (Iqbal et al., 2024) identified various approaches to integrating AI into Integrated Modular Avionics (IMA):

1. The model is computed within an ARINC653 partition (see case study ADIMA 2.4).
2. The model runs on dedicated ML hardware inside the IMA module.
3. The model runs on dedicated ML hardware outside the IMA module.

For option (1), there is no need to modify the development approach or re-certify, as it utilizes existing avionics-grade hardware. However, this comes with limitations in computational power, memory, and processing time[3]. Option (2) requires

3 this measures the ability of the model to correctly identify all relevant instances within a dataset, true positives/(true positives + false negatives)

modifying the avionics hardware to include an additional component and re-certifying it against the highest targeted DAL, typically DAL A for IMA hardware. Certification for highly complex ML-focused hardware presents significant challenges, including multi-core processors, non-deterministic hardware features (e.g., speculative branch execution), new memory technologies, and the reluctance of hardware vendors to disclose the internal architecture of these devices. Option (3) allows for a reduction in the DAL, requiring compliance only with the DAL of the specific component utilizing the ML model. The device resides outside the original, highly safety-critical module, connecting via CAN or another bus, enabling the avionics module to function as a gatekeeper to mitigate potential damage from a faulty ML hardware module.

4.1.5 Probabilistic systems considerations

Some systems in the aviation industry have probabilistic requirements, such as the Global Positioning System (GPS) and the Instrument Landing System (ILS). According to AC 20-138, GPS accuracy is based on measuring distances to satellites and their known locations, with a required route/terminal accuracy of 0.124 nautical miles at a 95% probability for Visual Flight Rules (VFR) use. This probabilistic approach acknowledges that GPS is affected by equipment and geometric factors, with some errors mitigated through mathematical models. GPS equipment for VFR use does not need TSO-C129 authorization but must not interfere with other required systems or create hazards. It is crucial that the system warns the user promptly when GPS data should not be used. However, pilots have reported issues with GPS not indicating system failures, exacerbated by increasing GPS jamming in certain areas.

CS-AWO Issue 2 sets standards for certifying aircraft and equipment for safe operation under various weather conditions, focusing on precise landing capabilities in low visibility. These standards include acceptable probabilities for exceeding limits, such as a touchdown within 60 m of the runway threshold, targeting a probability of 10^{-6}. Compliance requires a flight test program that covers a range of weights, center-of-gravity positions, xLS ground facility characteristics, airplane configurations, and wind speeds, involving three ILS facilities with evenly distributed approaches. Due to the impracticality of numerous approaches, a 90% confidence level is required. Compliance can be shown using the *continuous method* or the *pass or fail method*. The *continuous method* involves 30 approaches and records deviations to calculate success probabilities, while the *pass or fail method*, suitable when recording equipment is not possible, requires at least 46 successful approaches, with approval based on approach success.

These two systems illustrate different approaches aviation has used to manage performance uncertainty caused by external conditions. This is akin to machine learning, where data can deviate from intended distributions due to external factors. The aviation domain needs to offer similar guidance for demonstrating sufficiency through testing or ensuring that ML failures do not impact safety, as is done with these systems.

4.1.6 Certification process consideration

Certification authorities provide regulations to ensure safety and reliability in aviation. However, the means of compliance with these regulations are defined through standards developed by a consensus of international experts, including applicants, tool vendors, OEMs, and academia. These standards offer established frameworks that outline objectives and recommended activities to achieve the necessary assurance for compliance. In addition, applicants can propose novel means of compliance, though this introduces significant risks, as there is no certainty of acceptance by the certification authorities. The standards are typically based on best development practices, yet the aviation sector lacks sufficient practices to fully inform these standards, creating a “chicken and egg” dilemma where certification is required before flight, but practices need to be established through flying experience. Aviation faces more stringent safety demands than the automotive industry, which has addressed this through beta versions. Furthermore, developing these standards can span several years, posing a challenge to keep pace with the rapid advancements in machine learning, raising the risk that standards may become obsolete by the time they are published.

Once a system is certified, software cannot be updated without triggering a re-certification, which can be time-consuming and costly. In contrast, continuous development is vital for machine learning to ensure models remain accurate, relevant, and robust in dynamic environments. It allows regular updates to manage changing data patterns, improve performance through hyperparameter tuning and algorithm advancements, and address model degradation due to concept drift. In the context of certification, standards should also provide guidance for continuous re-certification of the system, balancing model efficiency—particularly for elements that impact system safety—and thorough system verification.

4.1.7 Tool qualification considerations

DO-330 (Tool Qualification Considerations) addresses the qualification of tools used in the software development process (RTCAInc, 2011). The primary focus of DO-330 includes code generation, verification and validation support, and other automation activities related to development artifacts. According to DO-330, the qualification of a tool is necessary when DO-178C processes are eliminated, reduced, or automated using software tools without subsequent verification of the tool’s outputRTCA, Inc. (2011a). To determine the Tool Qualification Level (TQL), the developer first needs to define the Tool Criteria:

Criteria 1: *A tool whose output is part of the resulting software and thus could insert an error* (RTCAInc, 2011).
Criteria 2: A tool that automates verification processes such that faulty output could lead to the elimination or reduction of development or verification processes.
Criteria 3: *A tool that, within the scope of its intended use, could fail to detect an error* (RTCAInc, 2011).

Together with the target resulting software DAL, the TQL is assigned a value from 1 to 5, which determines the objectives that the tool must cover.

In non-AI-based applications, the number of tools requiring qualification is typically small: code generators, validation tools, and verification tools. The development of ML applications introduces complexity due to the increased number of tools involved. Tools for managing datasets, training algorithms, validating models, and deploying them on target hardware — along with any conversion processes that may be required — are essential for machine learning research. Developers also utilize tools to detect outliers and create tests.

There are difficulties in classifying these tools using the criteria from above. For instance, a data management tool may not generate software output, automate processes, or detect errors, yet it plays a crucial role in storing data and associated verification results. The obvious solution here is to verify the output of such a tool - because this is the clear line to not qualifying a tool. However, this approach becomes futile when dealing with thousands of datasets - or requires a qualified *checker*-tool.

4.2 Applying W-process and overarching properties to ADIMA

Currently, the EASA guidelines do not fully encompass automated AI-enabled systems. Since their primary focus is on human-machine interaction, a significant gap exists for systems like the case study. We are unsure if these are applicable to this type of function, especially for AI levels. Notably, the number of objectives to be covered does not vary significantly with the associated DAL. The gap between DAL A and DAL D applications is small. There is no difference between DAL A and B, and the differences between DAL B and C primarily stem from independence requirements.

From a technical perspective, we lack reliable methods to fulfill specific objectives, such as **DM-08**: *The applicant should perform a data verification step to confirm the appropriateness of the defined ODD and of the data sets used for the training, validation, and verification of the ML model.* While verifying that the testing data remains independent during development and has not been used during training is clear, finding sufficient data to entirely reflect the ODD and still be appropriate for actual deployment is significantly more challenging. This difficulty shows especially when asking, *how much is enough?*. We believe the guidelines provide a valid step towards integrating AI applications in aircraft, especially those that directly support the pilot by providing reasoning and explanation for AI-based system suggestions. However, these guidelines cannot be fully applied to systems as small as ADIMA (Section 2.4). In addition, the standard should be evaluated for AI-enabled systems where AI is a minor contributor and does not directly affect aircraft controls or interact with the flight crew.

Unlike traditional standards, OPs provide applicants with significant flexibility to propose novel means of compliance. While this flexibility can increase the complexity of the certification process—since the authority must first approve the Means of Compliance (MoC)—it is particularly valuable in emerging technologies like machine learning, where no established certification standards exist. OPs enable a collaborative agreement between applicants and authorities, allowing for an MoC tailored to a specific system. The OPRA framework helps identify gaps in machine learning technologies, such as verification, that limit their use in safety-critical applications. Technological gaps of the ML technology are discussed in detail in the next section. In contrast to the EASA-W, OPs can propose a tailored set of evidence (equivalent to an objective) specific to the application. Additionally, OPs do not require the creation of a separate item for the machine learning component, enabling the use of advanced MoCs only where needed while still ensuring compliance with the well-established DO-178C standard for the rest of the software.

4.3 Limitations of ML/AI technology

In addition to the limitations of current certification standards and processes highlighted in the previous section, the development of ML technologies still needs to offer methods that fully guarantee system safety. Therefore, this section delves into the inherent challenges in ML technology that must first be addressed by the research community. By overcoming these technological challenges, regulatory authorities can develop a robust and feasible means of compliance for ML systems supported by validated methods.

4.3.1 Specification

Specifying ML systems presents a significant challenge due to the complex nature of their behavior, influenced by the data and learning algorithms, including hyperparameters, initialization, and loss functions. Currently, there are no established best practices for describing both functional and non-functional requirements for ML, particularly in capturing the learning process, model architecture, and handling issues like dead nodes (Ahmed et al., 2023). The ODD has emerged as a critical tool for specifying data requirements for data acquisition and testing. While covering the entire ODD can be extensive, leveraging corner and edge cases to limit the parameter space is a strategy. However, this can only be done automatically within a limited number of dimensions. Therefore, eliciting the ODD from domain experts can help reduce the parameter space and identify relevant cases. However, this remains challenging due to a need for clear terminology, methods to assess completeness, identification of all relevant scenarios, and undocumented assumptions (Heyn et al., 2022). Furthermore, data requirements should also capture implementation limitations, such as the differing corner cases for computer vision algorithms using RGB *versus* infrared images. Due to these challenges in defining the ODD, operational data is used to validate and update the specification, showing reliable results in autonomous processes. Nevertheless, this process does not work well for off-nominal conditions.

4.3.2 Data dependence

The performance of an ML model heavily depends on the quality of its training data. The model reflects the training data, i.e., any bias, incompleteness, or noise will lead to unreliable predictions. Ensuring that the training data is representative of real-world scenarios is crucial but often difficult to achieve. Verification methods must account for these data-related issues, adding another layer of complexity to the process. In addition to the traceability gap between requirements and datasets, the numerous data transformations, model iterations, and deployment environments of complex ML systems further complicate traceability. ML model development encompasses multiple steps, including data preprocessing, feature engineering, model training, and continuous learning. Tracking the lineage and transformations of data and models through these phases is crucial for verification.

This lack of traceability makes it challenging to reproduce results and identify the sources of errors or biases in the model.

4.3.3 Verification

Requirement-based testing (RBT) is a method where test cases are derived directly from requirement specifications to ensure that the system meets its specified requirements. In both classical software and machine learning systems, a significant portion of the effort is focused on identifying relevant test cases. The literature outlines various techniques to create scenarios, enhance training data for robustness, and validate data properties. However, a universally accepted method to ensure the sufficiency and quality of these test cases is still lacking. As neural networks grow in size and complexity, the number of parameters and possible states increases exponentially, necessitating significant computational resources for verification. Furthermore, formal verification methods are often constrained by the types of network activations and architectures supported. Techniques designed for specific activation functions, such as ReLU, may not extend efficiently to others, like sigmoid or tanh, without significant compromises (EASA and Daedalean, 2020; Katz et al., 2017; Grimm et al., 2024). Additionally, while ML models typically provide statistical guarantees, these may not always be adequate for safety-critical applications, where even a single misclassification can lead to severe consequences.

Structural testing, also known as white-box testing, involves examining the internal structure of software to identify unintended behaviors. Metrics like MC/DC help guide the extent of testing required. Neuron coverage has proven ineffective for coverage methods and thus implies ML being a black box. However, explainability local methods, such as LIME, SHAP, sensitivity analysis, and saliency maps, can partially bring light to the black box. These methods focus on understanding and debugging, often providing only local explanations rather than driving comprehensive testing (Bodria et al., 2023). Global XAI methods, such as surrogate models that approximate the behavior of a complex system, feature importance metrics across entire datasets, and inherently interpretable models like decision trees and rule-based algorithms, offer valuable insights into the overall behavior of AI models, making them potentially useful for certification. However, these methods can sometimes result in overgeneralized explanations as they may not accurately reveal the nuances of how the model behaves in specific edge cases or extreme scenarios. This means the explanations could overlook important details about the model's response under unusual conditions, which may lead to unintended behaviors. In addition, another challenge with feature importance lies in its potential unreliability, as the ranking of features can vary significantly depending on the model's complexity (Akhiat et al., 2021). This variability can introduce inconsistencies, making it difficult to establish a stable and trustworthy basis for certification, where consistent and interpretable feature rankings are crucial.

Data-driven approaches could be used to identify unintended behaviors, specifically by detecting data within the ODD but outside the training data distribution. Despite these advancements, the methods still require human input to classify undesired behavior.

During model training, developers must balance achieving high accuracy on training data, which can lead to overfitting, and ensuring the model can generalize well to unseen data. Generalization depends on several factors, including the quality of the data, the training process, and the test data used. Although theoretical definitions of generalization bounds and their direct connections to model complexity metrics can be misleading (MLEAP Consortium, 2024), practical methods to enhance generalization exist. The efficacy of these methods needs to be evaluated and implemented on a case-by-case basis, requiring further research.

4.3.4 Safety-driven development

Currently, training in machine learning is primarily focused on minimizing loss, which is directly related to performance. Training methods should emphasize safety and robustness for safety-critical applications rather than just performance. Henne et al. (2020) have analyzed the relationship between training properties and safety performance.

Additionally, assessing the reliability of ML estimations is crucial for ensuring safety. Confidence scores aim to provide a reliability measurement for ML models. However, they have been shown to be inadequate for unseen data (Hendrycks et al., 2021). Therefore, new uncertainty quantification methods have been developed to address this issue (Grewal et al., 2024). Techniques such as Monte Carlo dropout and deep ensembles look promising in providing reliable uncertainty measurements.

Due to the limitations of ML verification methods, run-time assurance has emerged as an approach to ensure the safety and reliability of a system during its operation by continuously monitoring its performance and behavior in real-time to detect and mitigate potential safety violations. However, to ensure safety, the safety and performance requirements must be clearly measurable, and there must be a less complex and easily verifiable safe backup function. This has proven challenging for complex systems Cofer et al. (2020).

4.3.5 Non-determinism

Non-determinism refers to the property of certain algorithms where the result can vary despite the same input and internal conditions. This behavior is often due to system abstractions (e.g., race conditions in operating systems that can cause applications to behave non-deterministically). Verification of these algorithms is challenging due to their unpredictable outcomes, making reproduction and test coverage difficult. For this reason, safety-critical systems often strive to avoid or tightly control such behavior. Non-determinism appears in ML in the following aspects 1) Inference: Small variations in input data, such as imperceptible noise in images, can lead to different neural network outputs. Therefore, monitoring and logging system behavior is key to identifying non-deterministic behavior. Sensitivity analysis, robustness checks, and formal verification techniques can help evaluate these effects. 2) Hardware Execution: GPUs used for complex ML algorithms can introduce non-determinism due to parallel processing and hardware-specific optimizations. 3) Development: Variations during training due to setting random seeds, controlling data shuffling, and specifying architectures. Even with controlled randomness, variations in model performance must be rigorously evaluated to ensure they don't compromise system security. Once the sources and effects of non-determinism are identified, formal verification, testing, and analysis must be used to demonstrate that the system's non-determinism is predictable and remains within an acceptable range.

4.3.6 Transfer learning and COTS

Deep learning requires vast amounts of labeled data to achieve high performance, making data an invaluable resource and often costly due to manual labeling. Sharing data can reduce the competitive advantage, although withholding it can limit safety due to the necessity of extensive datasets. The lack of standardized benchmarks for assessing verification methods in academia presents significant challenges. Without these benchmarks, it is difficult to objectively compare the performance of various methods, which hinders progress and innovation in the field as researchers struggle to identify the most effective techniques and understand their limitations. Transfer learning, a technique where a pre-trained model developed for a specific task serves as a starting point for a related task, offers advantages such as reduced data requirements and training time. However, it also presents risks, including privacy concerns, potential negative transfer that can degrade performance, and challenges in measuring transferability and interpretability across domains (Zhuang et al., 2021). The reusability of models and transfer learning will be crucial for developing Commercial Off-The-Shelf (COTS) systems. Like classical software, it is vital to determine the type of information that needs to flow to the system integrator to ensure training assumptions are respected during integration and how much operational design domain information from the system integrator needs to be communicated to the COTS developer for training. Whether this process will resemble current software integration practices or require more extensive interaction remains uncertain.

4.3.7 Continuous development

MLOps (Kreuzberger et al., 2023) aims to facilitate the creation of machine learning products by leveraging automated workflows that integrate development and operations. Beyond improving efficiency, MLOps addresses safety concerns related to model performance degradation, which can result from insufficient training data and the dynamic, ever-changing system environment. MLOps requires continuous monitoring, training, and evaluation to mitigate these issues. Identifying model degradation, often through detecting model drift, remains challenging (Bayram et al., 2022). Traditional point-in-time certifications with manual audits are insufficient for MLOps due to the frequent changes in ML systems (Granlund et al., 2021). Therefore, it is essential to establish requirements and regulations for when retraining should occur and the extent of verification needed before and after deployment.

5 Conclusion

The integration of AI in aerospace offers significant potential but requires rigorous attention to safety and certification. Our literature review highlights AI's pivotal role in predictive maintenance, anomaly detection, engine health monitoring, fuel consumption optimization, collision avoidance, aircraft trajectory prediction, and autonomous flight operations. We observed complex AI methods in high-criticality systems and simpler AI methods in low-criticality systems.

To advance the certification of AI-enabled aviation systems, we assessed existing aviation certification standards and examined their applicability to ML technologies. Current standards must expand to address AI's unique challenges, especially its probabilistic behavior. Standards must identify new failure causes, assess the applicability of architectural mechanisms to ML, integrate ML-specific methods for specification, traceability, and verification, and accommodate ML-accelerating hardware. Furthermore, the certification process must become more agile, supporting iterative updates to ensure continuous safety assurance.

We applied current certification efforts for certifying ML-aviation systems, such as EASA concept paper 2.0 and Overarching Properties (OPs), to ADIMA, an AI-enabled peripherical detection system used for DAL A to D systems. Although we found alignment with several objectives, the concept paper focuses too heavily on pilot support, emphasizing explainability and human factors, which is not relevant or applicable for all applications. The created OP argument revealed critical defeaters due to missing technological methods.

Drawing from our experience applying certification efforts and analysis of ML technology, we identified the need for further research in critical areas: specifying intended behavior for validation, improving verification methods (particularly testing sufficiency), developing training methods and model architectures that prioritize safety over performance, and validating knowledge transfer in the face of data limitations.

Future work should focus on creating adaptive standards and developing inherently safer, more transparent ML methods.

Data availability statement

The datasets presented in this study can be found in online repositories. The names of the repository/repositories and accession number(s) can be found below: https://github.com/ils-stuttgart/adima.

Author contributions

BL: Conceptualization, Data curation, Investigation, Methodology, Project administration, Resources, Software, Supervision, Validation, Visualization, Writing–original draft, Writing–review and editing. YA: Conceptualization, Data curation, Investigation, Methodology, Project administration, Resources, Software, Supervision, Validation, Visualization, Writing–original draft, Writing–review and editing. ZD: Conceptualization, Data curation, Investigation, Methodology, Project administration, Resources, Software, Supervision, Validation, Visualization, Writing–original draft, Writing–review and editing.

Funding

The author(s) declare that no financial support was received for the research, authorship, and/or publication of this article.

Conflict of interest

The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Publisher's note

References

Ahmed, S., Imtiaz, S. M., Khairunnesa, S. S., Cruz, B. D., and Rajan, H. (2023). "Design by contract for deep learning APIs," in *Proceedings of the 31st ACM joint European software engineering conference and symposium on the foundations of software engineering* (New York, NY, USA: Association for Computing Machinery), 94–106. ESEC/FSE 2023. doi:10.1145/3611643.3616247

Akhiat, Y., Manzali, Y., Chahhou, M., and Zinedine, A. (2021). A new noisy random forest based method for feature selection. *Cybern. Inf. Technol.* 21, 10–28. doi:10.2478/cait-2021-0016

Alzubaidi, L., Zhang, J., Humaidi, A. J., Al-Dujaili, A., Duan, Y., Al-Shamma, O., et al. (2021). Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. *J. big Data* 8, 53–74. doi:10.1186/s40537-021-00444-8

Amit, R. A., and Mohan, C. K. (2021). A robust airport runway detection network based on R-CNN using remote sensing images. *IEEE Aerosp. Electron. Syst. Mag.* 36, 4–20. doi:10.1109/maes.2021.3088477

Angelov, P. P., Soares, E. A., Jiang, R., Arnold, N. I., and Atkinson, P. M. (2021). Explainable artificial intelligence: an analytical review. *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.* 11, e1424. doi:10.1002/widm.1424

Ayhan, S., Pesce, J., Comitz, P., Sweet, D., Bliesner, S., and Gerberick, G. (2013). "Predictive analytics with aviation big data," in 2013 Integrated Communications, Navigation and Surveillance Conference (ICNS), Herndon, VA, United States, 22-25 April 2013, 1–13. doi:10.1109/ICNSurv.2013.6548556

Baptista, M. L., Henriques, E. M., and Prendinger, H. (2021). Classification prognostics approaches in aviation. *Measurement* 182, 109756. doi:10.1016/j.measurement.2021.109756

Baumann, S., and Klingauf, U. (2020). Modeling of aircraft fuel consumption using machine learning algorithms. *CEAS Aeronautical J.* 11, 277–287. doi:10.1007/s13272-019-00422-0

Bauranov, A., and Rakas, J. (2021). Designing airspace for urban air mobility: a review of concepts and approaches. *Prog. Aerosp. Sci.* 125, 100726. doi:10.1016/j.paerosci.2021.100726

Bayram, F., Ahmed, B. S., and Kassler, A. (2022). From concept drift to model degradation: an overview on performance-aware drift detectors. *Knowledge-Based Syst.* 245, 108632. doi:10.1016/j.knosys.2022.108632

Bejarano, C., Vázquez, A. L. R., Colomer, A., Cantero, J., Ferreira, A., Moens, L., et al. (2022). Harvis: dynamic rerouting assistant using deep learning techniques for single pilot operations (spo). *Transp. Res. Procedia* 66, 262–269. doi:10.1016/j.trpro.2022.12.026

Biannic, J., Hardier, G., Roos, C., Seren, C., and Verdier, L. (2016). Surrogate models for aircraft flight control: some off-line and embedded applications. *Aerosp. Lab.*–1. doi:10.12762/2016.AL12-14

Bodria, F., Giannotti, F., Guidotti, R., Naretto, F., Pedreschi, D., and Rinzivillo, S. (2023). Benchmarking and survey of explanation methods for black box models. *Data Min. Knowl. Discov.* 37, 1719–1778. doi:10.1007/s10618-023-00933-9

Boujamza, A., and Elhaq, S. L. (2022). Attention-based LSTM for remaining useful life estimation of aircraft engines. *IFAC-PapersOnLine* 55, 450–455. doi:10.1016/j.ifacol.2022.07.353

Chen, C.-J., Huang, C.-N., and Yang, S.-M. (2023). Application of deep learning to multivariate aviation weather forecasting by long short-term memory. *J. Intelligent and Fuzzy Syst.* 44, 4987–4997. doi:10.3233/jifs-223183

Chen, M., and Hu, Y. (2024). An image-based runway detection method for fixed-wing aircraft based on deep neural network. *IET Image Process.* 18, 1939–1949. doi:10.1049/ipr2.13087

Chen, T., and Guestrin, C. (2016). "XGBoost: a scalable tree boosting system," in Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, USA, August 13-17, 2016, 785–794. doi:10.1145/2939672.2939785

Chin, H.-J., Payan, A., Johnson, C., and Mavris, D. N. (2019). Phases of flight identification for rotorcraft operations. *AIAA Scitech 2019 Forum*, 0139. doi:10.2514/6.2019-0139

Choi, H.-C., Deng, C., and Hwang, I. (2021). Hybrid machine learning and estimation-based flight trajectory prediction in terminal airspace. *IEEE Access* 9, 151186–151197. doi:10.1109/access.2021.3126117

Cofer, D., Amundson, I., Sattigeri, R., Passi, A., Boggs, C., Smith, E., et al. (2020). "Run-time assurance for learning-enabled systems," in *NASA formal methods. NFM 2020. Lecture Notes in Computer Science*. Editors R. Lee, S. Jha, A. Mavridou, and D. Giannakopoulou (Cham: Springer International Publishing) 12229. doi:10.1007/978-3-030-55754-6_21

Corrado, S. J., Puranik, T. G., Pinon-Fischer, O. J., Mavris, D., Rose, R., Williams, J., et al. (2021). Deep autoencoder for anomaly detection in terminal airspace operations. *AIAA Aviat. 2021 Forum*, 2405. doi:10.2514/6.2021-2405

Das, S., Matthews, B. L., Srivastava, A. N., and Oza, N. C. (2010). "Multiple kernel learning for heterogeneous anomaly detection: algorithm and aviation safety case study," in *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, 47–56. doi:10.1145/1835804.1835813

da Silva, F. C., Grinet, M., and Silva, A. R. (2022). A machine learning approach to forecasting turbofan engine health using real flight data. *AIAA SCITECH 2022 Forum*, 0491. doi:10.2514/6.2022-0491

Dave, G., Choudhary, G., Sihag, V., You, I., and Choo, K.-K. R. (2022). Cyber security challenges in aviation communication, navigation, and surveillance. *Comput. and Secur.* 112, 102516. doi:10.1016/j.cose.2021.102516

Daw, Z., Beecher, S., Holloway, M., and Graydon, M. (2021). Overarching properties as means of compliance: an industrial case study. *IEEE/AIAA 40th Digital Avionics Systems Conference (DASC)*, San Antonio, TX, United States 5, 1–10. doi:10.1109/DASC52595.2021.9594298

De Giorgi, M. G., Campilongo, S., and Ficarella, A. (2018). A diagnostics tool for aero-engines health monitoring using machine learning technique. *Energy Procedia* 148, 860–867. doi:10.1016/j.egypro.2018.08.109

Delseny, H., Gabreau, C., Gauffriau, A., Beaudouin, B., Ponsolle, L., Alecu, L., et al. (2021). White paper machine learning in certified systems. doi:10.48550/arXiv.2103.10529

Ducoffe, M., Carrere, M., Féliers, L., Gauffriau, A., Mussot, V., Pagetti, C., et al. (2023). Lard–landing approach runway detection–dataset for vision based landing. *arXiv preprint arXiv:2304.09938*. doi:10.48550/arXiv.2304.09938

Ducoffe, M., Povéda, G., Galametz, A., Boumazouza, R., Martin, M.-C., Baris, J., et al. (2024). "Surrogate neural networks local stability for aircraft predictive maintenance", in *Formal Methods for Industrial Critical Systems*. Cham: Springer Nature Switzerland, 245–258.

EASA and Daedalean (2020). Concepts of Design Assurance for Neural Networks (CoDANN). Available at: https://www.easa.europa.eu/en/document-library/general-publications/concepts-design-assurance-neural-networks-codann.

EASA and Daedalean (2024). Concepts of design assurance for neural networks (CoDANN) II with appendix B.

Espinosa Barcenas, O. U., Quijada Pioquinto, J. G., Kurkina, E., and Lukyanov, O. (2023). Surrogate aerodynamic wing modeling based on a multilayer perceptron. *Aerospace* 10, 149. doi:10.3390/aerospace10020149

European Union Aviation Safety Agency (EASA) (2024). EASA Artificial Intelligence (AI) Concept Paper Issue 2: Guidance for Level 1&2 machine learning applications. *General Publ. Issue 02.* Available at: https://www.easa.europa.eu/en/document-library/general-publications/easa-artificial-intelligence-concept-paper-issue-2.

Fadlullah, Z. M., Tang, F., Mao, B., Kato, N., Akashi, O., Inoue, T., et al. (2017). State-of-the-art deep learning: evolving machine intelligence toward tomorrow's intelligent network traffic control systems. *IEEE Commun. Surv. and Tutorials* 19, 2432–2455. doi:10.1109/COMST.2017.2707140

Fala, N., Georgalis, G., and Arzamani, N. (2023). Study on machine learning methods for general aviation flight phase identification. *J. Aerosp. Inf. Syst.* 20, 636–647. doi:10.2514/1.i011246

Food and Drug Administration (2021). Good machine learning practice for medical device development: guiding principles. *U. S. Food Drug Adm.* Available at: https://www.fda.gov/medical-devices/software-medical-device-samd/good-machine-learning-practice-medical-device-development-guiding-principles.

Gaikwad, P., Mukhopadhyay, A., Muraleedharan, A., Mitra, M., and Biswas, P. (2023). Developing a computer vision based system for autonomous taxiing of aircraft. *Aviation* 27, 248–258. doi:10.3846/aviation.2023.20588

Garcia, A. B., Babiceanu, R. F., and Seker, R. (2021). "Artificial intelligence and machine learning approaches for aviation cybersecurity: an overview," in 2021 Integrated Communications Navigation and Surveillance Conference (ICNS), Dulles, VA, United States, 19-23 April 2021, 1–8. doi:10.1109/ICNS52807.2021.9441594

Granlund, T., Stirbu, V., and Mikkonen, T. (2021). Towards regulatory-compliant MLOps: oravizio's journey from a machine learning experiment to a deployed certified medical product. *SN Comput. Sci.* 2, 342. doi:10.1007/s42979-021-00726-1

Grewal, R., Tonella, P., and Stocco, A. (2024). "Predicting safety misbehaviours in autonomous driving systems using uncertainty quantification," in *2024 IEEE Conference on Software Testing, Verification and Validation (ICST)*, Toronto, ON, Canada, 28, 70, 81. doi:10.1109/icst60714.2024.00016

Grimm, D., Tollner, D., Kraus, D., Török, Á., Sax, E., and Szalay, Z. (2024). A numerical verification method for multi-class feed-forward neural networks. *Expert Syst. Appl.* 247, 123345. doi:10.1016/j.eswa.2024.123345

Giovanni, B., Martino, F. B., Anna, C., Calin, C., Luuk, V. D., and Peter, D. L. (2021). Neural network based runway landing guidance for general aviation autoland. doi:10.21949/1524481

Han, P., Liu, Y., and Cheng, Z. (2021). "Airport runway detection based on a combination of complex convolution and ResNet for PolSAR images," in *2021 SAR in Big Data Era (BIGSARDATA)*, Nanjing, China, 1–4. doi:10.1109/BIGSARDATA53212.2021.9574366

Heidari, A., Werthen-Brabants, L., Dhaene, T., Couckuyt, I., Onur, C., van Gils, P., et al. (2024). Data-driven surrogate modeling for the flammability reduction system. *AIAA SCITECH 2024 Forum*, 0785. doi:10.2514/6.2024-0785

Hendrycks, D., Zhao, K., Basart, S., Steinhardt, J., and Song, D. (2021). "Natural adversarial examples," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 15262–15271. Available at: https://arxiv.org/abs/1907.07174.

Henne, M., Schwaiger, A., Roscher, K., and Weiss, G. (2020). Benchmarking uncertainty estimation methods for deep learning with safety-related metrics. *SafeAI@AAAI*, 83–90.

Heyn, H.-M., Subbiah, P., Linder, J., Knauss, E., and Eriksson, O. (2022). "Setting AI in context: a case study on defining the context and operational design domain for automated driving," in *Requirements engineering: foundation for software quality*. Editors V. Gervasi and A. Vogelsang (Cham: Springer International Publishing), 199–215. doi:10.1007/978-3-030-98464-9_16

Huang, C., and Cheng, X. (2022). Estimation of aircraft fuel consumption by modeling flight data from avionics systems. *J. Air Transp. Manag.* 99, 102181. doi:10.1016/j.jairtraman.2022.102181

Huo, J., Keung, K. L., Lee, C. K. M., Ng, K. K. H., and Li, K. (2020). "The prediction of flight delay: big data-driven machine learning approach," in 2020 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM), USA, 14-17 Dec. 2020, 190–194. doi:10.1109/IEEM45057.2020.9309919

Intelligence, A., and Learning, M. (2021). Based software as a medical device (samd) action plan. *Food Drug Adm.*, 2021–2106.

SAE International (2023). *Guidelines for development of civil aircraft and systems*. b edn. Warrendale, PA: SAE International. ARP 4754B.

Iqbal, Z., Lehmann, M., Luettig, B., Ayyildiz, R., Bobrzik, T., Vardanega, T., et al. (2024). "Introducing ML to IMA technology – system perspective," in *Proceedings of the 43rd digital avionics systems conference (DASC)*, San diego, CA.

Jalawkhan, M. S., and Mustafa, T. K. (2021). "Anomaly detection in flight data using the naïve bayes classifier," in 2021 7th International Conference on Contemporary Information Technology and Mathematics (ICCITM), Mosul, Iraq, 26–30. doi:10.1109/ICCITM53167.2021.9677655

Janakiraman, V. M., and Nielsen, D. (2016). "Anomaly detection in aviation data using extreme learning machines," in 2016 international joint conference on neural networks (IJCNN), Vancouver, BC, Canada, 1993–2000. doi:10.1109/IJCNN.2016.7727444

Jia, P., Chen, H., Zhang, L., and Han, D. (2022). Attention-LSTM based prediction model for aircraft 4-D trajectory. *Sci. Rep.* 12, 15533. doi:10.1038/s41598-022-19794-1

Jiangyan, Z., Ma, J., and Wu, J. (2024). A regularized constrained two-stream convolution augmented transformer for aircraft engine remaining useful life prediction. *Eng. Appl. Artif. Intell.* 133, 108161. doi:10.1016/j.engappai.2024.108161

Jiuxiang, G., Ma, L., Shahroudy, A., Shuai, B., et al. (2018). Recent advances in convolutional neural networks. *Pattern Recognit.* 77, 354. doi:10.1016/j.patcog.2017.10.013

Kato, N., Fadlullah, Z. M., Tang, F., Mao, B., Tani, S., Okamura, A., et al. (2019). Optimizing space-air-ground integrated networks by artificial intelligence. *IEEE Wirel. Commun.* 26, 140–147. doi:10.1109/MWC.2018.1800365

Katz, G., Barrett, C., Dill, D. L., Julian, K., and Kochenderfer, M. J. (2017). "Reluplex: an efficient smt solver for verifying deep neural networks," in *Computer Aided Verification. CAV 2017. Lecture Notes in Computer Science*, Editor R. Majumdar and V. Kunčak (Springer, Cham), 10426. doi:10.1007/978-3-319-63387-9_5

Kefalas, M., Baratchi, M., Apostolidis, A., van den Herik, D., and Bäck, T. (2021). "Automated machine learning for remaining useful life estimation of aircraft engines," in *2021 IEEE international conference on prognostics and health management (ICPHM)*, Detroit (Romulus), MI, Unite States, 1–9. doi:10.1109/ICPHM51084.2021.9486549

Koch, W., Mancuso, R., West, R., and Bestavros, A. (2019). Reinforcement learning for uav attitude control. *ACM Trans. Cyber-Physical Syst.* 3, 1–21. doi:10.1145/3301273

Kong, Y., Zhang, X., and Mahadevan, S. (2022). Bayesian deep learning for aircraft hard landing safety assessment. *IEEE Trans. intelligent Transp. Syst.* 23, 17062–17076. doi:10.1109/tits.2022.3162566

Korvesis, P. (2017). Machine learning for predictive maintenance in aviation. *Theses, Univ. Paris Saclay (COmUE).*

Kreuzberger, D., Kühl, N., and Hirschl, S. (2023). Machine learning operations (MLOps): overview, definition, and architecture. *IEEE Access* 11, 31866–31879. Conference Name: IEEE Access. doi:10.1109/ACCESS.2023.3262138

Lazzara, M., Chevalier, M., Colombo, M., Garcia, J. G., Lapeyre, C., and Teste, O. (2022). Surrogate modelling for an aircraft dynamic landing loads simulation using an lstm autoencoder-based dimensionality reduction approach. *Aerosp. Sci. Technol.* 126, 107629. doi:10.1016/j.ast.2022.107629

Lee, H., Li, G., Rai, A., and Chattopadhyay, A. (2020). Real-time anomaly detection framework using a support vector regression for the safety monitoring of commercial aircraft. *Adv. Eng. Inf.* 44, 101071. doi:10.1016/j.aei.2020.101071

Leško, J., Andoga, R., Bréda, R., Hlinková, M., and Fözö, L. (2023). Flight phase classification for small unmanned aerial vehicles. *Aviation* 27, 75–85. doi:10.3846/aviation.2023.18909

Li, X., Ding, Q., and Sun, J.-Q. (2018). Remaining useful life estimation in prognostics using deep convolution neural networks. *Reliab. Eng. and Syst. Saf.* 172, 1–11. doi:10.1016/j.ress.2017.11.021

Lindemann, B., Müller, T., Vietz, H., Jazdi, N., and Weyrich, M. (2021). A survey on long short-term memory networks for time series prediction. *Procedia CIRP* 99, 650–655. doi:10.1016/j.procir.2021.03.088

Lingrui, L., and Xin, W. (2024). Towards smart aviation with sustainable development: artificial intelligence insights into the airline and advanced air mobility industries. *Decis. Support Syst. Sustain. Comput.*, 187–204doi. doi:10.1016/B978-0-443-23597-9.00009-3

Liu, C., and Ferrari, S. (2019). Vision-guided planning and control for autonomous taxiing via convolutional neural networks. *AIAA Scitech 2019 Forum* 50, 0928. doi:10.2514/6.2019-0928

Liu, L., Tian, L., Kang, Z., and Wan, T. (2023). Spacecraft anomaly detection with attention temporal convolution networks. *Neural Comput. Appl.* 35, 9753–9761. doi:10.1007/s00521-023-08213-9

Liu, Y., Liu, Y., Hansen, M., Pozdnukhov, A., and Zhang, D. (2019). Using machine learning to analyze air traffic management actions: ground delay program case study. *Transp. Res. Part E Logist. Transp. Rev.* 131, 80–95. doi:10.1016/j.tre.2019.09.012

Loh, H. W., Ooi, C. P., Seoni, S., Barua, P. D., Molinari, F., and Acharya, U. R. (2022). Application of explainable artificial intelligence for healthcare: a systematic review of the last decade (2011–2022). *Comput. Methods Programs Biomed.* 226, 107161. doi:10.1016/j.cmpb.2022.107161

Lu, M., Peng, W., He, M., and Teng, Y. (2021). Flight delay prediction using gradient boosting machine learning classifiers. *J. Quantum Comput.* 3, 1–12. doi:10.32604/jqc.2021.016315

Luettig, B., and Annighoefer, B. (2023). "Using autoencoders to identify aged, faulty and unknown peripherals in the adaptive ima system," in 2023 IEEE/AIAA 42nd Digital Avionics Systems Conference (DASC), Barcelona, Spain, October 5, 2023, 1–9. doi:10.1109/DASC58513.2023.10311122

Luettig, B., Dallmann, J., and Annighoefer, B. (2022). "ADIMA: automatic configuration by peripheral detection and adaptive distributed task execution for integrated modular avionics platforms," in 2022 IEEE/AIAA 41st Digital Avionics Systems Conference (DASC), USA, 18-22 Sept. 2022, 1–10. doi:10.1109/DASC55683.2022.9925885

Luithardt, P. (2017). *Formale Validierung eines Verfahrens zur konsistenten Master/Shadow-Festlegung in einem verteilten, nicht uhrensynchronen Avioniksystem*. phdthesis: Universität Stuttgart.

Ma, L., and Tian, S. (2020). A hybrid CNN-LSTM model for aircraft 4D trajectory prediction. *IEEE access* 8, 134668–134680. doi:10.1109/access.2020.3010963

Mathew, V., Toby, T., Singh, V., Rao, B. M., and Kumar, M. G. (2017). "Prediction of remaining useful lifetime (rul) of turbofan engine using machine learning," in 2017 IEEE international conference on circuits and systems (ICCS), Thiruvananthapuram, India, 306–311. doi:10.1109/ICCS1.2017.8326010

MLEAP Consortium (2024). EASA research – machine learning application approval (MLEAP) final report. *European Union Aviation Safety Agency*. Available at: https://www.easa.europa.eu/en/research-projects/machine-learning-application-approval.

Mohsan, S. A. H., Othman, N. Q. H., Li, Y., Alsharif, M. H., and Khan, M. A. (2023). Unmanned aerial vehicles (uavs): practical aspects, applications, open challenges, security issues, and future trends. *Intell. Serv. Robot.* 16, 109–137. doi:10.1007/s11370-022-00452-4

Muñoz-Esparza, D., Sharman, R. D., and Deierling, W. (2020). Aviation turbulence forecasting at upper levels with machine learning techniques based on regression trees. *J. Appl. Meteorology Climatol.* 59, 1883–1899. doi:10.1175/jamc-d-20-0116.1

Nanyonga, A., Wasswa, H., and Wild, G. (2023). "Phase of flight classification in aviation safety using lstm, gru, and bilstm: a case study with asn dataset," in 2023 International Conference on High Performance Big Data and Intelligent Systems (HDIS) (Los Alamitos, CA, United States: IEEE), 24–28. doi:10.1109/HDIS60872.2023.10499521

NASA/TM–2019–220292 (2019). Understanding the overarching properties. *Tech. Rep.*

Nasoulis, C. P., Mantziou, S., Gkoutzamanis, V. G., and Kalfas, A. I. (2023). Test rig design considerations to detect volatile organic compounds in aircraft cabins. *J. Phys. Conf. Ser.* 2511, 012012. doi:10.1088/1742-6596/2511/1/012012

Oehling, J., and Barry, D. J. (2019). Using machine learning methods in airline flight data monitoring to generate new operational safety knowledge from existing data. *Saf. Sci.* 114, 89–104. doi:10.1016/j.ssci.2018.12.018

Ouahouah, S., Bagaa, M., Prados-Garzon, J., and Taleb, T. (2022). Deep-reinforcement-learning-based collision avoidance in uav environment. *IEEE Internet Things J.* 9, 4015–4030. doi:10.1109/JIOT.2021.3118949

Que, Z., Liu, Y., Guo, C., Niu, X., Zhu, Y., and Luk, W. (2019). "Real-time anomaly detection for flight testing using autoencoder and LSTM," in 2019 International Conference on Field-Programmable Technology (ICFPT), Tianjin, China, 379–382. doi:10.1109/ICFPT47387.2019.00072

Rebollo, J. J., and Balakrishnan, H. (2014). Characterization and prediction of air traffic delays. *Transp. Res. Part C Emerg. Technol.* 44, 231–241. doi:10.1016/j.trc.2014.04.007

Rohani, A. S., Puranik, T. G., and Kalyanam, K. M. (2023). "Machine learning approach for aircraft performance model parameter estimation for trajectory prediction applications," in *2023 IEEE/AIAA 42nd Digital Avionics Systems Conference (DASC)*, Barcelona, Spain, 1–9. doi:10.1109/DASC58513.2023.10311271

RTCA, Inc (2000). Design assurance guidance for airborne electronic hardware. *Tech. Rep. DO-254, Radio Tech. Comm. Aeronautics*.

RTCA, Inc (2011). RTCA DO-330: software tool qualification considerations. *Tech. Rep.* DO-330.

SAE International (1996). *Guidelines and methods for conducting the safety assessment process on civil airborne systems and equipment*. Warrendale, PA, USA: Tech. Rep. ARP4761, Society of Automotive Engineers.

Samek, W., Montavon, G., Lapuschkin, S., Anders, C. J., and Müller, K.-R. (2021). Explaining deep neural networks and beyond: a review of methods and applications. *Proc. IEEE* 109, 247–278. doi:10.1109/JPROC.2021.3060483

Sarker, I. H. (2021). Deep learning: a comprehensive overview on techniques, taxonomy, applications and research directions. *SN Comput. Sci.* 2, 420. doi:10.1007/s42979-021-00815-1

Schimpf, N., Wang, Z., Li, S., Knoblock, E. J., Li, H., and Apaza, R. D. (2023). A generalized approach to aircraft trajectory prediction via supervised deep learning. *IEEE Access* 11, 116183–116195. doi:10.1109/access.2023.3325053

Shakhatreh, H., Sawalmeh, A. H., Al-Fuqaha, A., Dou, Z., Almaita, E., Khalil, I., et al. (2019). Unmanned aerial vehicles (UAVs): a survey on civil applications and key research challenges. *IEEE Access* 7, 48572–48634. doi:10.1109/access.2019.2909530

Shi, Z., Xu, M., and Pan, Q. (2020). 4-D flight trajectory prediction with constrained lstm network. *IEEE Trans. intelligent Transp. Syst.* 22, 7242–7255. doi:10.1109/tits.2020.3004807

Shi, Z., Xu, M., Pan, Q., Yan, B., and Zhang, H. (2018). "LSTM-based flight trajectory prediction," in 2018 International joint conference on neural networks (IJCNN), Brazil, Jul. 08 - 13, 2018 (IEEE), 1–8. doi:10.1109/IJCNN.2018.8489734

Sommerwerk, K., Michels, B., Lindhorst, K., Haupt, M., and Horst, P. (2016). Application of efficient surrogate modeling to aeroelastic analyses of an aircraft wing. *Aerosp. Sci. Technol.* 55, 314–323. doi:10.1016/j.ast.2016.06.011

Sridhar, B., Chatterji, G. B., and Evans, A. D. (2020). Lessons learned in the application of machine learning techniques to air traffic management. *AIAA Aviat. 2020 FORUM*, 2882. doi:10.2514/6.2020-2882

Stanton, I., Munir, K., Ikram, A., and El-Bakry, M. (2023). Predictive maintenance analytics and implementation for aircraft: challenges and opportunities. *Syst. Eng.* 26, 216–237. doi:10.1002/sys.21651

Tang, C., and Lai, Y.-C. (2020). "Deep reinforcement learning automatic landing control of fixed-wing aircraft using deep deterministic policy gradient," in 2020 International Conference on Unmanned Aircraft Systems (ICUAS), Athens, Greece, 1–9. doi:10.1109/ICUAS48674.2020.9213987

Topal, B., Çarkacıoğlu, L., and Töreyin, B. U. (2023). "Machine learning prediction based UI for aircraft cockpit," in *2023 14th International Conference on Electrical and Electronics Engineering (ELECO)*, Bursa, Turkiye, 1–5. doi:10.1109/ELECO60389.2023.10416080

US Department of Transportation (2022). Summary report: standing general order on crash reporting for level 2 advanced driver assistance systems. *Tech. Rep. Natl. Highw. Traffic Saf. Adm.*

Vollert, S., and Theissler, A. (2021). "Challenges of machine learning-based RUL prognosis: a review on NASA's C-MAPSS data set," in *2021 26th IEEE international conference on emerging technologies and factory automation (ETFA)*, Vasteras, Sweden, 1–8. doi:10.1109/ETFA45728.2021.9613682

Wang, H., Zhang, Z., Li, X., Deng, X., and Jiang, W. (2023). Comprehensive dynamic structure graph neural network for aero-engine remaining useful life prediction. *IEEE Trans. Instrum. Meas.* 72, 1–16. doi:10.1109/tim.2023.3322481

Woo, J., and Kim, N. (2020). Collision avoidance for an unmanned surface vehicle using deep reinforcement learning. *Ocean. Eng.* 199, 107001. doi:10.1016/j.oceaneng.2020.107001

Yang, K., Bi, M., Liu, Y., and Zhang, Y. (2019). "LSTM-based deep learning model for civil aircraft position and attitude prediction approach," in *2019 Chinese control conference (CCC)*, Guangzhou, China, 8689–8694. doi:10.23919/ChiCC.2019.8865874

Yasin, J. N., Mohamed, S. A., Haghbayan, M.-H., Heikkonen, J., Tenhunen, H., and Plosila, J. (2020). Unmanned aerial vehicles (UAVs): collision avoidance systems and approaches. *IEEE access* 8, 105139–105155. doi:10.1109/access.2020.3000064

Yondo, R., Bobrowski, K., Andrés, E., and Valero, E. (2019). A review of surrogate modeling techniques for aerodynamic analysis and optimization: current limitations and future challenges in industry. *Adv. Evol. deterministic methods Des. Optim. control Eng. Sci.*, 19–33. doi:10.1007/978-3-319-89988-6_2

Zeng, W., Quan, Z., Zhao, Z., Xie, C., and Lu, X. (2020). A deep learning approach for aircraft trajectory prediction in terminal airspace. *IEEE Access* 8, 151250–151266. doi:10.1109/access.2020.3016289

Zhang, Q., Mott, J. H., Johnson, M. E., and Springer, J. A. (2021). Development of a reliable method for general aviation flight phase identification. *IEEE Trans. Intelligent Transp. Syst.* 23, 11729–11738. doi:10.1109/tits.2021.3106774

Zhang, W., Jin, F., Zhang, G., Zhao, B., and Hou, Y. (2019). "Aero-engine remaining useful life estimation based on 1-dimensional FCN-LSTM neural networks," in 2019 Chinese Control Conference (CCC), Guangzhou, China, 4913–4918. doi:10.23919/ChiCC.2019.8866118

Zhao, W., Li, L., Alam, S., and Wang, Y. (2021a). An incremental clustering method for anomaly detection in flight data. *Transp. Res. Part C Emerg. Technol.* 132, 103406. doi:10.1016/j.trc.2021.103406

Zhao, Y., Guo, J., Bai, C., and Zheng, H. (2021b). Reinforcement learning-based collision avoidance guidance algorithm for fixed-wing uavs. *Complexity* 2021, 8818013. doi:10.1155/2021/8818013

Zhong, J., Zhang, Y., Wang, J., Luo, C., and Miao, Q. (2021). Unmanned aerial vehicle flight data anomaly detection and recovery prediction based on spatio-temporal correlation. *IEEE Trans. Reliab.* 71, 457–468. doi:10.1109/tr.2021.3134369

Zhuang, F., Qi, Z., Duan, K., Xi, D., Zhu, Y., Zhu, H., et al. (2021). A comprehensive survey on transfer learning. *Proc. IEEE* 109, 43–76. doi:10.1109/JPROC.2020.3004555