Toward Design Assurance of Machine-Learning Airborne Systems

Konstantin Dmitriev *
Institute of Flight Systems Dynamics, Technische Universität München, Germany
Email: konstantin.dmitriev@tum.de

Johann Schumann†
KBR/Wyle, NASA Ames Research Center, Moffett Field, CA, 94035
Email: johann.m.schumann@nasa.gov

Florian Holzapfel‡
Institute of Flight Systems Dynamics, Technische Universität München, Germany
Email: florian.holzapfel@tum.de

In recent years, Artificial Intelligence (AI) systems, enabled by Machine Learning (ML) technology, have demonstrated impressive progress and provides historic opportunities for the aviation industry. However, several key aspects of ML technology are not compatible with existing design assurance standards and make certification problematic. In this paper, we present a case study of a visual system with a Deep Neural Network (DNN) intended to detect and identify airport runway signs. Different use cases and variants of this system exhibit different levels of criticality ranging from design assurance level (DAL) D to B. We use the case study to illustrate the challenges of certification according to the current standards, such as DO-178C. We present the system design, data generation, training, and verification in detail and describe how the design assurance objectives can be met for a DAL D variant of the system. We also discuss gaps and potential approaches for the higher design assurance levels.

I. Introduction

Artifical Intelligence (AI) has gained tremendously in popularity during the past few years and has found its way into many automotive and aerospace applications. In particular, Deep Neural Networks (DNN) provide major opportunities for computer vision and natural speech processing applications. The European Aviation Safety Agency (EASA) has recently published a roadmap [1] envisioning an increasing use of artificial intelligence and, in particular, machine learning (ML) systems in commercial aviation in the near future. Relevant application areas include pilot assistance and autonomous operation using vision-based automation of important phases of the flight like landing, taxiing, and takeoff, as well as preventive maintenance, and air traffic management.

All these applications have in common that they are more or less safety critical; failure in their operation can range from minor to catastrophe. Therefore, such AI systems need to undergo aviation certification. However, several key aspects of machine learning technology are not compatible with the objectives of avionics certification standards [2]. Since 2019, the aviation industry is cooperating in the scope of the EUROCAE WG-114 and SAE G-34 joint working group with a view to developing a new industry standard to address the concerns of ML systems certification. However, the rigorous process of a new industry standard creation is expected to take several years. Similarly, the AI roadmap published by the European Aviation Safety Agency [1] does not anticipate certification of ML-based system earlier than 2025. Lack of updated regulations is a great limitation to the potential advantages of ML applications in aviation. In this work we propose a solution for addressing this limitation.

This paper presents a case study of a hypothetical neural-network based component for the detection and interpretation of airport and runway signs during taxiing of an aircraft. Our Runway Sign Classifier (RSC) uses an on-board camera and a Deep Neural Network (DNN) for sign detection, classification, and reading. With this case study, we will discuss certifiability of low-criticality (design assurance level (DAL) D) ML systems based on the existing industry certification

*Research Associate, Institute of Flight System Dynamics, Boltzmannstrasse 15, 85748 Garching, Germany
†Chief Scientist Computational Science, M/S 269-3, Moffett Field, CA 94035, AIAA Member
‡Professor, Institute of Flight System Dynamics, Boltzmannstrasse 15, 85748 Garching, Germany, AIAA Associate Fellow

standards and show that the DAL D objectives of the DO-178C standard [3] can be achieved if the proposed ML development workflow is followed. For the higher design assurance levels, we discuss the additional mitigation means.

The rest of this paper is structured as follows: In Section II we introduce our case study, the RSC system and its various use cases. A custom development workflow and requirements for this ML component are presented and discussed in detail. Additional subsections cover design-related topics like the RSC architecture, the acquisition and analysis of training data, the training process, and verification activities. Section III focuses on the analysis of certification compliance for an RSC instantiation of DAL D and includes the discussion of the higher assurance levels. In Section IV, we discuss related work, and Section V summarizes the paper and concludes.

II. Case Study

A. System Overview

Throughout this paper, we will illustrate the development of a safety critical ML-based system using a small, yet a realistic example. Taxiing on a busy airport can be challenging. In addition to watching out for other aircraft or vehicles on the tarmac, the pilot must read the runway and taxiway markers and signs to understand the current aircraft position and heading (Figure 1). For an unmanned aircraft (UAS), the on-board flight software must be able to recognize the signs and potentially react upon them.

Fig. 1 Typical airport sign

Locating a (traffic) sign within a camera image and classifying shape and content is a task, where convolutional deep neural networks are typically used. Numerous approaches and systems have been developed for the detection and classification of road signs in the automotive domain (see [4, 5] for an overview). Similarly, [6] describes a system for the automatic detection of runway signs on airports.

In the following, we will introduce our hypothetical AI system, the Runway Sign Classifier (RSC). The RSC receives visual images from a forward-facing camera on the aircraft and uses a pretrained DNN for detection and classification of runway signs at an airport. For the purpose of this paper, we consider three hypothetical support systems that are based upon the same DNN-based sign recognition architecture (Figure 2):

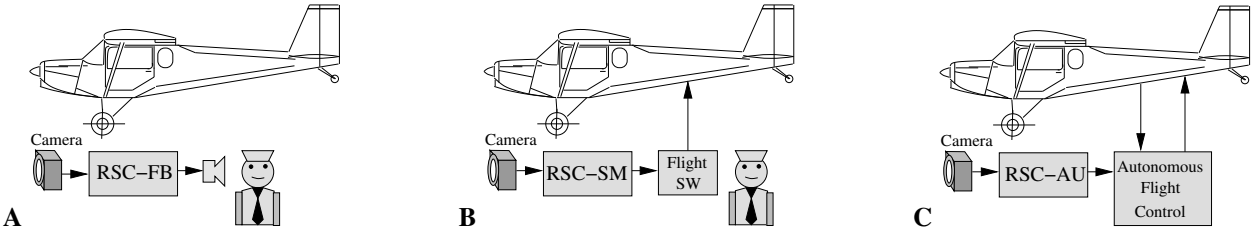

Fig. 2 High-level architecture of the different RSC variants. A: RAC-FB, B: RSC-SM, C: RSC-AU.

RSC-FB (RSC - Flight Bag). This variant of the RSC is used as a support and convenience system for the pilot: whenever a runway sign is detected, an acoustic signal is emitted and the sign's content will be read out to inform the pilot. Thus RSC-FB could increase the awareness of the pilot towards the runway markers. However, a failure of the system would only have minimal safety impact, since the pilot would still be able to visually see the markers and the pilot is fully in charge of controlling the aircraft. Therefore, we can assume that the RSC-FB has a minor potential impact on safety and the low design assurance level (DAL D) can be justified. In a similar setting, the RSC-FB would augment the electronic flight bag.

RSC-SM (RSC - Safety Monitor). In this case, we consider the RSC-SM as a safety monitor. The RSC-SM is trained to detect important safety signs, like a "do not enter" sign. If the aircraft is moving within a certain velocity range, the RSC-SM would first issue a cockpit annunciation and, upon non-reaction, would initiate an automatic brake of the aircraft. RSC-SM is designed to be active in taxiing mode only and, if necessary, can be turned off. Thus, annunciation and brake action can be overridden by the pilot. Hence, this system could be classified as DAL C.

RSC-AU (RSC - Autonomy). This variant of RSC could be deployed on a fully autonomous aircraft, where the detected runway markers are used to identify the location of the aircraft and to control the aircraft during taxi. In this scenario, the location and sign information would directly feed into the autonomous decision system for controlling the aircraft and for avoiding any runway incursion. Such a system would be highly safety critical and assigned with DAL B or DAL A depending on the aircraft category.

We use the various use cases of RSC with different criticality as a driving example for our analysis to illustrate the certification issues identified during the standards' analysis and evaluate the potential alternative verification and validation methods.

B. System Requirements and Architecture

Each variant of our RSC system needs to be defined by requirements. Since all variants use (by design) the same DNN component, there is a substantial overlap. In this paper, we focus on certification topics and issues regarding the DNN component. We start with a number of system-level requirements for RSC, which are shown in Table 1. Requirements concerning the specific operations and constraints of each variant, as well as the integration into the aircraft flight system are assumed to be defined. Here, these requirements are only used to determine the DAL of the system. Numbers given in the table have been selected arbitrarily.

ID	Description
RSC-1	The RSC shall detect type and content of the mandatory instruction (red background) airport signs and information (yellow background) airport signs according to FAA AC 150/5340-18G Standards for Airport Signs Systems.
RSC-2	The RSC shall operate under normal lighting conditions and with clear visibility.
RSC-3	The RSC shall recognize a sign if it is less than 75m from the aircraft.
RSC-4	The RSC shall recognize signs to the left and right of the AC located within a maximum angle of 20 degrees.
RSC-5	The RSC shall detect at least 3 signs on an image.
RSC-6	The RSC shall limit the detection latency to 500ms.
RSC-7	The RSC shall have a precision of sign detection and content identification of at least 95%.
RSC-8	The RSC shall use a forward-facing camera with a frame rate of at least 10Hz.

Table 1 Selected system-level requirements for the RSC

ID	Description
RSC-DNN-1a	The RSC DNN shall process the input 128×128 RGB image to detect the presence of the mandatory instruction signs (red background) and information (yellow background) airport runway signs according to FAA AC 150/5340-18G Standards for Airport Signs Systems.
RSC-DNN-1b	The RSC DNN shall return the class (mandatory or information) of each detected sign and its bounding box position and size in pixels.
RSC-DNN-2	The RSC DNN shall operate under normal lighting conditions and with clear visibility
RSC-DNN-3	The RSC DNN shall recognize a sign if its height is in the range from 10 to 30 pixels
RSC-DNN-4	The RSC shall detect at least 3 signs on an image
RSC-DNN-5	The DNN shall process the image within 100 ms
RSC-DNN-6	The RSC DNN shall have an average sign detection precision above 95%. The model average precision on a dataset is calculated as the ratio of the total number of correct sign detections to the total ground-truth number of signs.

Table 2 Requirements for the RSC DNN component

From the system-level requirements in Table 1, the more detailed requirements for the DNN component (Table 2) can be derived. In contrast to the requirements in Table 1, which often concern typical properties of an entire aerospace systems, the requirements in Table 2 very specific with respect to the DNN component of the system and can use specific notations. These requirements will have a major impact on how the network architecture is defined and how the training and test data are selected. Other requirements, which concern the performance of the DNN are of probabilistic nature, as, for example, the DNN component must perform at least with a given detection precision (RSC-DNN-6). Depending on the actual machine learning tasks (e.g., detection, classification, or estimation), it might be necessary to use different and suitable ML performance metrics. For an overview of such metrics and their areas of application see, e.g., [7].

From these requirements, a detailed design of the RSC is developed; for details of the overall process see Section II.C below. Figure 3 shows the detailed architecture of the DNN component. The forward-facing camera provides an image of the current scenery. In our prototype implementation, we scale the image size to 128x128x3 pixels. These RGB image data are then input to the DNN. The DNN in our prototype implementation is based upon YOLO-V2 [8], a well-known DNN architecture for object detection. In RSC, this DNN has been trained prior to deployment and the weights are frozen. The training process is described in Section II.E. The DNN returns the class and bounding box (given as x and y coordinates in the image) of the detected signs as its output*. In a post-processing step, the input image is cropped around the bounding box to yield an image showing the sign only. That image is then processed with standard optical character recognition (OCR) software (e.g., tesseract†) to extract the text on the sign. In this paper, we will focus on the DNN component of RSC.

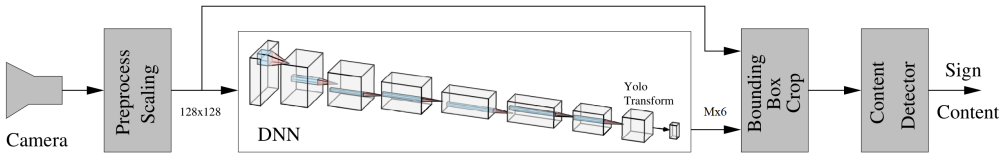

Fig. 3 Detailed architecture of the DNN core of RSC

Depending on the use case, the outputs of the DNN are fed into an acoustic annunciation system (RSC-FB) or the AC flight computer (RSC-SM and RSC-AU). Some post-processing of the signals might be necessary. This DNN architecture processes each camera image individually; no temporal dependencies are recognized. The RSC component is activated at a fixed rate, as specified in the requirements.

C. Custom ML Development Workflow

The goal of the case study was to satisfy for the DNN component of RSC the DAL D development and verification objectives of the design assurance standard. In order to achieve this goal, we applied the custom ML workflow proposed in [2]. We made the following adjustments to the complete workflow proposed in [2]:

- Testing of the trained ML model was performed on a PC platform. Integration testing with physical camera sensors and embedded hardware is not implemented because these activities have no significant differences against traditional methods for non-ML software/software and software/hardware integration, specifically in the context of a DAL D development.
- Training dataset was considered as ML model requirements and verified by review for accuracy, consistency and compliance with corresponding RSC DNN component requirements.
- Integration with non-ML software components was excluded because non-ML components can be developed and integrated using traditional methods and hence are of no specific interest in this case study.
- Planning, quality assurance, and certification liaisons processes were excluded because these processes are not impacted by the specifics of ML technology and can be implemented using conventional methods.
- The compliance analysis was focused on the DO-178C standard. As shown in [2], the design assurance principals of the other applicable standards, such as ARP-4574A and DO-254, are fundamentally the same and compliance can be demonstrated using the same approach.

The generic ML development process called "W-shaped process" shown in Figure 4 was originally introduced in [9] and further elaborated in [10] and [11]. That W-shape process provides an adaptation of the classical V-cycle

*DNN architecture in Figure 3 was visualized with `https://alexlenail.me/NN-SVG/AlexNet.html`

†`https://github.com/tesseract-ocr/tesseract`

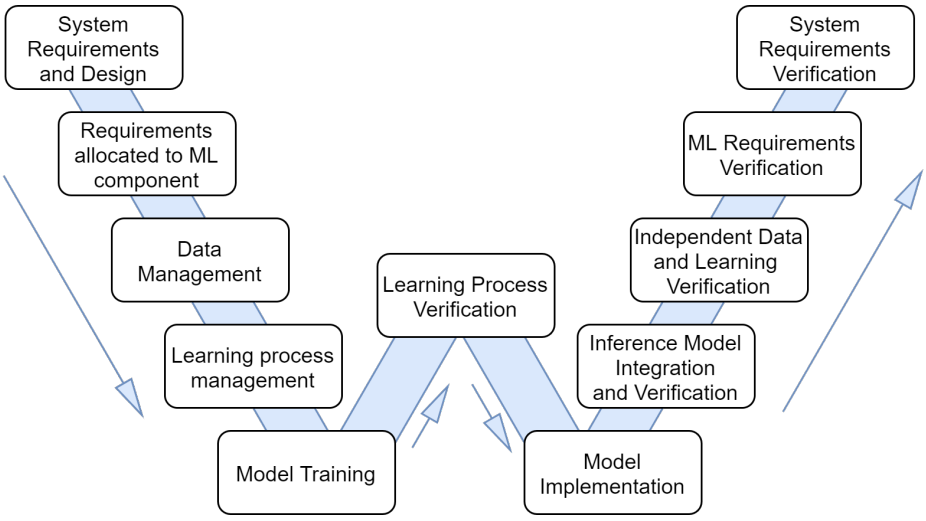

Fig. 4 W-shaped process for ML-specific development

(designed for non-ML systems and software) by including ML-specific development activities and in particular, the central "Learning process verification" phase to highlight the ability to verify ML-model by testing even before the actual software/hardware implementation is available for testing in the target hardware environment. Focusing on the ML-specific elements of the custom workflow [2], we implemented the following development, verification and validation activities:

- Requirements capture and system architecture definition II.B
- Data collection and verification II.D
- ML environment setup II.E
- ML model architecture definition and training II.E
- ML model verification II.F

Other non-essential aspects of the W-shaped process such as requirements and architecture verification, inference model implementation, integration, and verification are less specific to the ML context and can be implemented using the traditional approaches. Thus, they will not be addressed in the case study. The detailed discussion of the custom workflow activities with regards to the objectives of the DO-178C standard is provided in III below.

D. Data Collection and Preparation

The quality of data used for ML model training and testing is a key element to achieve good performance of an ML model. We have not found a publicly available dataset of airport signs images, and decided to primarily use synthetic data in order to reduce the lengthy process of real-world data collection and labeling. The synthetic dataset was supplemented with a smaller size dataset of real-world images, which was used for testing of the trained model performance under realistic conditions and afterwards for additional transfer learning of the model, which had been pre-trained on synthetic data.

1. Synthetic Dataset

For the generation of synthetic data, we developed a framework based on the FlightGear flight simulator controlled by a Simulink model and MATLAB script to automatically capture and label the data. The conceptual schema of the synthetic data generation framework is shown in Figure 5. The framework includes the following components:

- a MATLAB script, which fetches the list of airport sign coordinates and metadata from the FlightGear airport database and implements a loop through the fetched airport signs and the various viewpoint lateral offsets, distances, and heading angles randomized along the assumed taxiing trajectory. In each loop cycle, the script triggers the execution of the Simulink model and FlightGear simulator.
- a Simulink model, which queries FlightGear for the local ground elevation and transforms the viewpoint's relative position to geodetic viewpoint coordinates. The calculated viewpoint coordinates are packaged and sent to

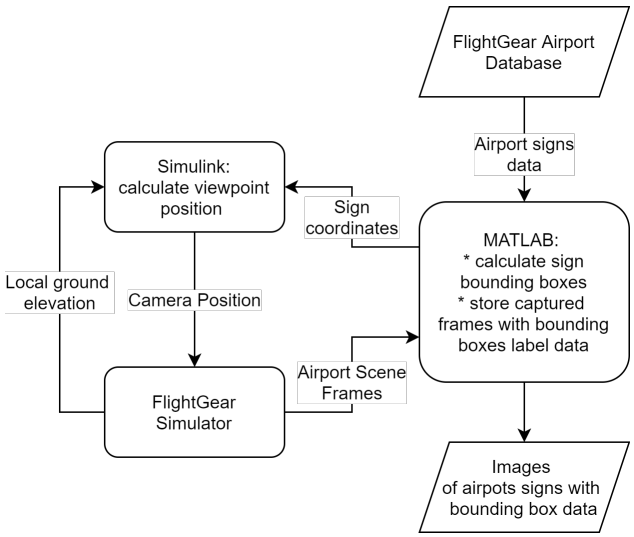

Fig. 5 Synthetic data generation framework

FlightGear using the MATLAB Aerospace Blockset.

- a MATLAB code that captures a frame of the FlightGear video stream for each viewpoint position, calculates the airport sign bounding box, and stores the information in the data structure.

Fig. 6 FlightGear labeled frame

The developed framework was executed on four airports (KSFO, KBOS, PANC, KSAN) representing different terrain features in different weather conditions and time of the day. Overall, about 1000 synthetic images have been generated resulting from the iterative trial/error process of training and performance evaluation. The final synthetic dataset includes selected 500 images. Figure 6 shows an example of the captured video frame with added bounded box and annotation of sign text specification (in yellow).

Our data generation system generates about 30 annotated synthetic images per minute on a medium performance PC platform (Intel i7-8550U, 16GB RAM) without GPU. The performance is mainly limited by the FlightGear rendering speed and can be improved using a GPU-enabled platform. However, the achieved performance was acceptable for the size of the necessary dataset. The synthetic dataset was randomly shuffled and divided into the following parts: 55% training set, 15% validation set and 30% test set.

2. Real-world Dataset

A relatively small dataset based on real-world data was created by manually labeling the actual images of airport signs. To do that we used publicly available YouTube videos such as airplane taxiing footage taken from the cockpit. We selected about 50 frames captured at different airports and under various operational conditions and annotated them using the ground truth labeling feature of MATLAB Computer Vision Toolbox[‡]. This dataset was used for testing the trained ML model in a realistic environment as described in II.F.

[‡]https://www.mathworks.com/help/vision/image-and-video-ground-truth-labeling.html

E. ML Model Training

Given the requirements and design considerations discussed in Section II.B, YOLO v2, a compact and well-studied object detection network [8], was selected as the DNN for our RSC case study. Training and testing of the ML model was implemented using the MATLAB R2021b Computer Vision Toolbox (CVT)[§] and the Deep Learning Toolbox[¶] that provides a rich framework for designing and testing of deep neural networks. Our RSC prototype DNN model was implemented using `yolov2ObjectDetector`[‖]. The resolution of the input layer was set to 128×128 pixels to meet the model requirements. The camera images were scaled accordingly. The selected resolution is intended to balance computational cost (high image resolution increases the computational cost) and view angle (low image resolution requires reducing the camera view angle).

The output of the DNN comprises the most likely bounding boxes of the detected runway signs. The yellow line around the sign in Figure 6 shows such a bounding box. The MATLAB YOLO object detector uses the concept of anchor boxes to represent this information. Based on the evaluation of our training dataset using MATLAB's `estimateAnchorBoxes`[**], anchor boxes of four different sizes were defined to model the most frequent sizes of the signs in the dataset. Thus, the DNN output layer was configured to represent anchor boxes of sizes 15×15, 15×40, 15×50, and 15×70, where the first number corresponds to the height of the sign in pixels.

Training was performed using the MATLAB CVT function `trainYOLOv2ObjectDetector` configured for a stochastic gradient descent with momentum (SGDM) search. The training algorithm loss function is defined as a sum of bounding box localisation loss (position and size), confidence loss, and classification loss with corresponding loss factor weights $K_1, \ldots, K_4$. In order to achieve the desired target model precision and to optimize training time, four selected hyper parameters were tuned using the exhaustive sweep feature of the MATLAB Experiment Manager[††]. We kept the default values of the other training parameters as specified in the SGDM training options class definition `TrainingOptionsSGDM`[‡‡]. The sweep resulted in setting the size of mini-batch to 13, the maximum number of epochs to 30, the learning rate to 0.001, and the loss factor weights to $K_1 = 6$, $K_2, \ldots, K_4 = 1$.

We performed several training iterations for the hyperparameters tuning and achieved an average precision above 99% on the validation dataset. Next, we implemented training/testing iterations gradually increasing the size of the synthetic dataset from 100 to 500 synthetic images to reach the desired model precision on the test dataset (see Section II.F).

F. Verification Activities

Verification activities are intended to detect errors that may have been made during development and can affect safety. The required depth of verification depends on the DAL, which is determined based on the criticality of the system. In our case study, we implemented verification activities to satisfy the DAL D verification objectives that are primarily focused on testing, supplemented by requirements reviews and test coverage analysis.

1. Testing

We implemented the following test methods to demonstrate compliance of the trained model with system requirements:

1) testing using the synthetic images dataset (synthetic testing) and
2) testing using the real-world images dataset (real-world testing).

Synthetic testing. The first method was implemented by exercising the trained model on the dataset of synthetic images. The test dataset included the normal range and robustness subsets of images. The normal range subset was a random selection of 30% images from the overall synthetic dataset (see Section II.D). For each normal input image, the model is expected to return the class of the detected sign(s) and its bounding box information (position and size). The sign is considered as correctly detected if the sign class matches the ground-truth class and the overlap of the detected bounding box with the ground truth bounding box is greater than 50% *(detection criteria)*. The model precision in normal range was calculated as the ratio of the total number of correct detections to the total ground-truth number of signs. The tests were considered as passed if the measured model precision is greater or equal to the specified precision *(normal range pass/fail criteria)*.

The robustness subset was created based on requirements by adding synthetic images to cover abnormal conditions

[§] `https://www.mathworks.com/help/vision/index.html`
[¶] `https://www.mathworks.com/help/deeplearning/index.html`
[‖] `https://www.mathworks.com/help/vision/ref/yolov2objectdetector.html`
[**] `https://www.mathworks.com/help/vision/ref/estimateanchorboxes.html`
[††] `https://www.mathworks.com/help/deeplearning/ref/experimentmanager-app.htmll`
[‡‡] `https://www.mathworks.com/help/deeplearning/ref/nnet.cnn.trainingoptionssgdm.html`

such as low lighting and poor visibility conditions (RCS-DNN-2) and images with objects containing text, which are not signs (RCS-DNN-1). For the robustness input image without a sign, the model was expected to return an empty detection array (no detection). For the robustness input image containing a sign but in poor visibility or low lighting, the model was expected to report no-detection, or detect the sign with the same detection criteria as for normal range tests. The model precision on the robustness subset was calculated as ratio of the number of images with reported non-detection or with correctly detected sign (if present) to the total number of images. The abnormal tests were considered as passed if the measured model precision is greater or equal to the specified precision *(robustness pass/fail criteria)*.

Several iterations of testing and training using the first method were implemented because the model accuracy on the test data was originally less than on the validation data. We reviewed detection failures to identify the root causes and implemented corrective actions to achieve the required ML model performance under all specified conditions. The most common issues that we identified were insufficient number of training data for a specific condition and errors in the synthetic data setup resulting in imprecise ground-truth bounding box data. When the required model precision on the synthetic dataset was achieved *(stop criteria)*, we transitioned to real-world testing.

Real-world testing. This method is intended to demonstrate the correct model behavior under real operating conditions. We applied the same pass/fail and stop criteria for both real-world and synthetic testing methods. The average precision on the combined real-world and synthetic test dataset exceeded 99%.

The goal of this method implementation was to demonstrate the concept rather than implementing a full-scale exhaustive testing on real-world data, which can be very expensive due to the manual process of data labeling. So, a relatively small real-world dataset of 50 images (about 30% of the final synthetic dataset, so that each requirement was covered by at least one test image) was used for the second phase testing. Although current certification practices do not explicitly prescribe the ratio of real-world vs. synthetic test data, we consider this topic as an important assurance aspect and will explore this further in future work.

Closed loop scenario-based testing. ML-based systems are typically characterized by a large input space. Scenario-based testing is discussed in [12] as a means to support conventional equivalence-class testing for complex ML-based systems. We have not included this testing method in this version of case study but are currently evaluating the scenario-based testing approach in combination with statistical methods such as Markov chain Monte Carlo simulations [13] and will address this topic in future work.

2. Reviews and Analyses

In addition to testing, DAL D verification activities include reviews and analyses of requirements and tests to check accuracy, consistency, traceability, and a sufficient coverage of requirements by tests. Given that the RSC requirements are expressed in the textual form, we performed the conventional peer review to check them for accuracy and consistency. Training dataset were assumed to be part of RSC requirements and were reviewed for accuracy, consistency and compliance with corresponding DNN component requirements. In order to trace test dataset to requirements and support the analysis of requirements coverage by tests, we grouped the test dataset images in the folders named to reflect the requirement ID covered by this test group. The groups of tests have been peer reviewed to confirm that test cases exist for each requirement and the criteria of normal and robustness testing is satisfied. Peer review can be facilitated by different means and tools such as interactive checklist ([14], [15]), which are specifically relevant for big projects with large numbers of requirements, but is not considered as necessary for our small example.

III. Compliance Analysis

A. DAL D objectives

In this section, we analyze the compliance of the workflow applied to the RSC DNN component development with the DO-178C objectives. Table 3 shows the list of DO-178C development and verification objectives applicable to the Level D with the corresponding compliance analysis. Planning, quality assurance, configuration management and certification liaisons activities are not analyzed, because, as shown in [2], they are not significantly affected by the distinctions of machine learning technology. Nevertheless, we carried out configuration identification, version control, and baselining to provide controllable configuration throughout the project.

Table 3 Level D Software Development and Verification Objectives

DO-178C Objectives		Software Levels					Compliance Analysis
		A	B	C	D	E	
1	High-level requirements (HLR) are developed.	x	x	x	x		In the proposed ML development workflow, the RSC DNN component requirements in conjunction with the training dataset serve as software high-level requirements [2]. They specifying functional, operational, and performance characteristics of RSC DNN component and training datasets, are developed (see II.B) to satisfy this objective.
2	Derived high-level requirements are defined and provided to the system processes, including the system safety assessment.	x	x	x	x		This case study is implemented independently from any aircraft development project and therefore has no upstream aircraft-level requirements or applicable regulations. Therefore, all developed requirements are identified as derived. The formal safety assessment process is out of the scope of this case study, nevertheless all requirements have been considered for the DAL evaluation as described in the system overview section II.A.
3	Software architecture is developed.	x	x	x	x		The software architecture outlines the software components and their interfaces to allow grouping of the software functions. As justified in [2], there is no need for breaking down the software module implementing an ML model into multiple architectural components and therefore this objective is not in the case study scope. Nonetheless, the architectural considerations may be required for the traditional software components or for the integration of ML software with the other software components and it can be implemented using the traditional software development practices.
4	Executable Object Code and Parameter Data Item (PDI) Files, if any, are produced and loaded in the target computer.	x	x	x	x		This objective is not included in the case study scope, because the activities for this objective do not differ between traditional and ML-based software components [2] and can be implemented using conventional software development practices.
5	High-level requirements comply with system requirements.	i	i	x	x		In the implemented ML development workflow, the RSC DNN component requirements serve as both system and high-level software requirements and therefore this objective is implicitly satisfied (Item 1 under the Table MB.A-3 of DO-331). Training dataset was verified for compliance with RSC DNN component requirements by peer review.
6	High-level requirements are accurate and consistent.	i	i	x	x		The developed RSC requirements and training dataset were verified for accuracy and consistency by peer review to satisfy this objective.
7	High-level requirements are traceable to system requirements.	x	x	x	x		Same as the Objective 5, given the RSC DNN component requirements serve as both system and high-level software requirements, this objective is implicitly satisfied. Traceability of training dataset elements to RSC DNN component requirements was verifeid by peer review.
8	Executable object code complies with high-level requirements.	x	x	x	x		Compliance with software high-level requirements (represented by RSC requirements), was demonstrated by requirements-based testing of ML executable object code on the host PC platform.
9	Executable object code is robust with high-level requirements.	x	x	x	x		The requirements-based testing activities include the abnormal range testing to demonstrate robustness with RSC requirements.
10	Executable object code is compatible with target computer.	x	x	x	x		Compliance with this objective can be demonstrated with conventional methods by exercising the selected subset of integration tests on a target hardware (e.g. hardware-in-the-loop testing in [14]). Since this activity has no specific considerations with respect to the ML development workflow, it is not included in the case study.
11	Test coverage of high-level requirements is achieved.	x	x	x	x		The test datasets have been peer reviewed to confirm that test cases exist for each requirement and the criteria of normal and robustness testing is satisfied.

B. DAL A-C objectives

Several DO-178C development and verification objectives for DAL C and higher levels, such as structural coverage analysis and low-level requirements verification, are problematic with respect to explainability and coverage challenges specific to ML technology [2]. In order to provide an acceptable level of confidence in safety, the gaps in the current standards must be mitigated by some alternative means. We identified and are currently evaluating the following mitigation means to address the known gaps:

- Reduction of design assurance level from C to D thorough the use of dissimilar redundant architecture of ML-based components [16].
- Neural network coverage analysis. Several research projects propose different DNN coverage metrics, such as

neuron activation and boundary coverage [17], [18] These novel metrics require further dependability analysis and mapping to the different design assurance levels.
- Scenario-based testing using Markov chain Monte Carlo simulation (see Section II.F).
- Run-time detection of abnormal operating conditions based on the neuron activation ranges. The conventional equivalence classes testing approach is not practical for ML-based systems due to the ML explainability challenge [2]. Instead, the neuron activation range recorded on normal range inputs might be used as a reference for detection of abnormal inputs. This method may require aggregation of multiple neuron states and careful layers selection and needs further evaluation.

IV. Related Work

A prototypical NN-based system for the detection and identification of airport signs and markings is described in [19]. The authors also use training and test data generated with the help of a flight simulator and their system is based on a convolutional NN for sign detection and identification, reaching a recognition rate of up to 85%. No details about the NN architecture is provided, however.

In general, the detection and recognition of signs in a complex environment has been studied quite carefully, in particular in the automotive domain (see, e.g., [5], for a review on earlier work). Such sign detection systems are not only important for self-driving cars or driver assistance systems, but can also be found as a convenience component in cars: the current speed limit is displayed in the cockpit based upon traffic signs that are recognized along the road.

For street signs, most of the published approaches use an extensive training data set, the German Traffic Sign Dataset (GTSRB) [20]. Numerous approaches and a variety of architectures to address this task can be found in the literature and on GitHub. As a typical example, [21] uses an approach based on combination of NN-based object detection (e.g., YOLO2) with feature extractors (like ResNet or MobileNet). [22] uses RetinaNet for the simultaneous detection and classification of traffic signs.

There are numerous approaches that address testing as well as validation and verification (V&V) of Neural Networks. [23, 24] provide overviews of existing approaches. On early works on the use of Neural Networks in safety-critical applications see [25]. The current standards for certification of safety-critical aerospace software, like DO-178C, do not address the specifics of machine-learning or neural networks. More recently, the European Aviation Safety Agency (EASA) has proposed a road-map for concepts and design assurance for Neural networks [9–11].

Run-time architectures (RTA) for the assurance of complex functions (like an AI or DNN component) have been defined, for example, in the ASTM3269 Runtime Assurance (RTA) architecture [26].

V. Conclusions and Future Work

This paper is concerned with the certification aspects of machine-learning systems or components for safety-critical airborne applications. In order to illustrate the approach for the mitigation of incompatibilities of ML technology with the aviation certification standards, we implemented a case study of a small but realistic ML-based system.

In the scope of this case study, we developed and verified the Runway Sign Classifier (RSC) – a hypothetical, vision-based application using a DNN to detect and recognize airport runway signs. Depending on the use case, this system can be categorized into DAL D, DAL C, or DAL B. As in all three variants the same DNN component is used, the RSC is an ideal case study for the analysis and discussion of certification objectives and tasks on the various DAL levels.

Based on the case study results, we performed an analysis of the DO-178C objectives applicable to DAL D and concluded that, using a custom ML development workflow, all necessary objectives can be met. For DAL C and higher assurance levels we identified the list of alternative mitigation means, which can be used to address the incompatibilities of ML technology with the current design assurance practices.

Future work will include detailed analysis and practical evaluation of alternative mitigation means for higher assurance levels that could be helpful to guide the development of customized workflows and targeted validation and verification techniques toward certification of safety-critical ML components in the aviation domain.

Acknowledgments. We would like to thank Shanza Zafar and Stephan Myschik for the careful review of this paper and helpful feedback.

References

[1] "Artificial Intelligence Roadmap. A human-centric approach to AI in aviation," Tech. rep., European Aviation Safety Agency, 2020.

[2] Dmitriev, K., Schumann, J., and Holzapfel, F., "Toward Certification of Machine-Learning Systems for Low Criticality Airborne Applications," *2021 AIAA/IEEE 40th Digital Avionics Systems Conference (DASC)*, IEEE, 2021, pp. 1–10.

[3] *Software Considerations in Airborne Systems and Equipment Certification*, RTCA DO-178C, Washington DC, 2011.

[4] Bahlmann, C., Zhu, Y., Ramesh, V., Pellkofer, M., and Koehler, T., "A system for traffic sign detection, tracking, and recognition using color, shape, and motion information," *IEEE Proceedings. Intelligent Vehicles Symposium, 2005.*, 2005, pp. 255–260.

[5] Mukhometzianov, R., and Wang, Y., "Review. Machine learning techniques for traffic sign detection," *CoRR*, Vol. abs/1712.04391, 2017. URL http://arxiv.org/abs/1712.04391.

[6] Chen, Z.-H., and Juang, J.-C., "Mitigation of Runway Incursions by Using a Convolutional Neural Network to Detect and Identify Airport Signs and Markings," *Sensors and Materials*, Vol. 31, No. 12, 2019, pp. 3947–3958.

[7] Hossin, M., and M.N, S., "A Review on Evaluation Metrics for Data Classification Evaluations," *International Journal of Data Mining & Knowledge Management Process*, Vol. 5, 2015, pp. 01–11. doi:10.5121/ijdkp.2015.5201.

[8] Redmon, J., and Farhadi, A., "YOLO9000: better, faster, stronger," *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7263–7271.

[9] "Concepts of Design Assurance for Neural Networks (CoDANN)," Tech. rep., European Aviation Safety Agency, 2020.

[10] "Report. Concepts of Design Assurance for Neural Networks (CoDANN) II," Tech. rep., European Aviation Safety Agency, 2021.

[11] "EASA Concept Paper: First usable guidance for Level 1 machine learning applications," Tech. rep., European Aviation Safety Agency, 2021.

[12] *Road vehicles — Safety of the intended functionality*, ISO/PAS 21448:2019, Geneva, Switzerland, 2019.

[13] Mishra, C., Schwaiger, F., Bähr, N. M., Sax, F., Kleser, M. A., Nagarajan, P., and Holzapfel, F., "Efficient Verification and Validation of Performance-Based Safety Requirements using Subset Simulation," *AIAA Scitech 2021 Forum*, 2021, p. 0072.

[14] Dmitriev, K., Zafar, S. A., Schmiechen, K., Lai, Y., Saleab, M., Nagarajan, P., Dollinger, D., Hochstrasser, M., Holzapfel, F., and Myschik, S., "A Lean and Highly-automated Model-Based Software Development Process Based on DO-178C/DO-331," *2020 AIAA/IEEE 39th Digital Avionics Systems Conference (DASC)*, IEEE, 2020, pp. 1–10.

[15] Schmiechen, K., Zafar, S. A., Dmitriev, K., Krammer, C., Maly, M., and Holzapfel, F., "A Requirements Management Template in Polarion for Model-Based Development of Airborne Systems." *Software Engineering (Satellite Events)*, 2021.

[16] Yan, F., "Effects of Architecture Design on Safety Level of Airborne Electronic Systems," *Applied Mechanics and Materials*, Vol. 236, Trans Tech Publ, 2012, pp. 862–868.

[17] Sun, Y., Huang, X., Kroening, D., Sharp, J., Hill, M., and Ashmore, R., "Structural test coverage criteria for deep neural networks," *ACM Transactions on Embedded Computing Systems (TECS)*, Vol. 18, No. 5s, 2019, pp. 1–23.

[18] Cheng, C.-H., Huang, C.-H., and Yasuoka, H., "Quantitative projection coverage for testing ml-enabled autonomous systems," *International Symposium on Automated Technology for Verification and Analysis*, Springer, 2018, pp. 126–142.

[19] Chen, Z., and Juang, J., "Mitigation of runway incursions by using a convolutional neural network to detect and identify airport signs and markings," *Sensors and Materials*, Vol. 31, No. 12, 2019, pp. 3947–3958. doi:10.18494/SAM.2019.2303, funding Information: This study was supported by the Ministry of Science and Technology (MOST), Taiwan, under grant no. MOST 105-2221-E-006-106-MY3.

[20] Houben, S., Stallkamp, J., Salmen, J., Schlipsing, M., and Igel, C., "Detection of Traffic Signs in Real-World Images: The German Traffic Sign Detection Benchmark," *International Joint Conference on Neural Networks*, 2013.

[21] Álvaro Arcos-García, Álvarez García, J. A., and Soria-Morillo, L. M., "Evaluation of deep neural networks for traffic sign detection systems," *Neurocomputing*, Vol. 316, 2018, pp. 332–344. doi:https://doi.org/10.1016/j.neucom.2018.08.009, URL https://www.sciencedirect.com/science/article/pii/S092523121830924X.

[22] Lin, T.-Y., Goyal, P., Girshick, R., He, K., and Dollár, P., "Focal Loss for Dense Object Detection," , 2018.

[23] Borg, M., Englund, C., Wnuk, K., Durán, B., Levandowski, C., Gao, S., Tan, Y., Kaijser, H., Lönn, H., and Törnqvist, J., "Safely Entering the Deep: A Review of Verification and Validation for Machine Learning and a Challenge Elicitation in the Automotive Industry," *CoRR*, Vol. abs/1812.05389, 2018. URL http://arxiv.org/abs/1812.05389.

[24] Urban, C., and Miné, A., "A Review of Formal Methods applied to Machine Learning," , 2021.

[25] Schumann, J., Gupta, P., and Liu, Y., *Application of Neural Networks in High Assurance Systems: A Survey*, Springer Berlin Heidelberg, Berlin, Heidelberg, 2010, pp. 1–19. doi:10.1007/978-3-642-10690-3_1, URL https://doi.org/10.1007/978-3-642-10690-3_1.

[26] Nagarajan, P., Kannan, S. K., Torens, C., Vukas, M. E., and Wilber, G. F., *ASTM F3269 - An Industry Standard on Run Time Assurance for Aircraft Systems*, 2021. doi:10.2514/6.2021-0525, URL https://arc.aiaa.org/doi/abs/10.2514/6.2021-0525.