Runway Sign Classifier: A DAL C Certifiable Machine Learning System

Konstantin Dmitriev
Technical University of Munich
Garching, Germany
konstantin.dmitriev@tum.de

Johann Schumann
KBR, NASA Ames Research Center
Moffett Field, CA
johann.m.schumann@nasa.gov

Islam Bostanov
Technical University of Munich
Garching, Germany
islam.bostanov@tum.de

Mostafa Abdelhamid
Technical University of Munich
Garching, Germany
mostafa.abdelhamid@tum.de

Florian Holzapfel
Technical University of Munich
Garching, Germany
florian.holzapfel@tum.de

***Abstract*—In recent years, the remarkable progress of Machine Learning (ML) technologies within the domain of Artificial Intelligence (AI) systems has presented unprecedented opportunities for the aviation industry, paving the way for further advancements in automation, including the potential for single pilot or fully autonomous operation of large commercial airplanes. However, ML technology faces major incompatibilities with existing airborne certification standards, such as ML model traceability and explainability issues or the inadequacy of traditional coverage metrics. Certification of ML-based airborne systems using current standards is problematic due to these challenges. This paper presents a case study of an airborne system utilizing a Deep Neural Network (DNN) for airport sign detection and classification. Building upon our previous work, which demonstrates compliance with Design Assurance Level (DAL) "D", we upgrade the system to meet the more stringent requirements of Design Assurance Level "C". To achieve DAL C, we employ an established architectural mitigation technique involving two redundant and dissimilar Deep Neural Networks. The application of novel ML-specific data management techniques further enhances this approach. This work is intended to illustrate how the certification challenges of ML-based systems can be addressed for medium criticality airborne applications.**

I. Introduction and Related Work

The remarkable progress of Machine Learning (ML) technologies in recent years has the potential to revolutionize aviation [1], [2]. Data-driven ML systems can implement highly complex cognitive functions such as vision and language processing that can enable single pilot or fully autonomous operation of large commercial airplanes, a level of automation not possible with traditional rule-based software systems [3]. However, ML technology encounters various inherent incompatibilities with existing airborne certification standards. These incompatibilities encompass challenges related to explainability, traceability, and implementation coverage [4]. As a result, the utilization of ML-based applications within the framework of existing airborne certification standards is currently impeded.

The industry, aviation authorities, and academia are actively engaged in collaborative efforts to develop new certification standards aimed at addressing the existing incompatibilities of ML technology with existing certification practices. The development of a new standard for airborne machine learning (ML) certification has been underway since 2019 through the collaborative efforts of the EUROCAE/SAE WG-114[1]/G-34[2] joint working group. The working group has published reports reflecting the intermediate results [4], [5]. The European Aviation Safety Agency (EASA) has released the guidance for ML applications [8], which includes anticipated objectives and means of compliance for the certification of safety-critical airborne systems based on machine learning. In collaboration with Daedalean AG, the U.S. Federal Aviation Administration (FAA) has released a research report on a neural network vision-based landing guidance system [6]. This report offers a practical assessment of the previously proposed W-shaped process for ML applications, as outlined in earlier Daedalean AG publications [1], [2]. Numerous ongoing research projects in academia are exploring various aspects of ML-based systems certification in different domains [7]–[11].

However, all currently available materials do not constitute final certification standards but rather anticipated guidance specific to systems utilizing machine learning technology. Due to the novelty and complexity of the subject matter, as well as the involvement of multiple stakeholders, the accomplishment and acceptance of new certification standards for ML may require several additional years. The absence of certification standards presents a fundamental constraint to the effective utilization of the advantages offered by ML technology within the aviation industry. To tackle these challenges, we proposed and implemented in our previous works [12], [13] a custom ML certification workflow for low-criticality (Design Assurance Level "D") [14] machine learning systems which is based on existing certification standards. In this work, we present an extension to our previous case study [13] of the Machine Learning Runway Sign Classifier (RSC) system designed to

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

[2]https://standardsworks.sae.org/standards-committees/g-34-artificial-intelligence-aviation

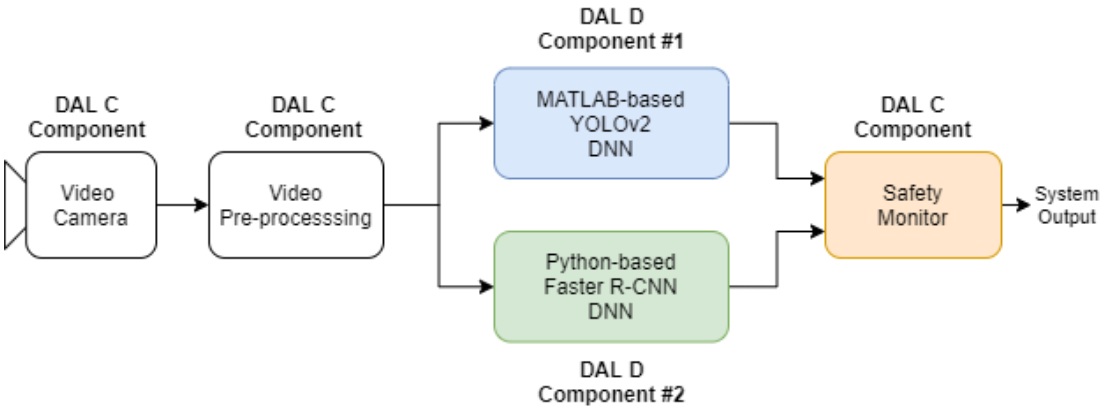

Fig. 1. Runway Sign Classifier DAL C System Architecture

detect and classify airport runway signs utilizing a computer vision deep neural network. Through this work we illustrate the implementation of the next version of our custom workflow addressing DAL C certification objectives as proposed in [15]. To achieve DAL C, we utilize the proven architectural mitigation approach in combination with novel ML-specific development and verification techniques. This work is intended to demonstrate the incremental certification approach from low DAL D to medium DAL C assurance level by the application of existing assurance practices.

The rest of this paper is structured as follows: Section II provides an overview of the RSC system under study including a discussion of its use cases, an overview of the system requirements and architecture, data management aspects, and an analysis of system component implementation, integration, verification and validation. Section III comprises an evaluation of the RSC system compliance with the DAL C certification objectives. Section IV provides a summary of the paper, explores future work, and presents the concluding remarks.

II. Case Study

In this work, we upgrade the ML-based Runway Sign Classifier system introduced in [13] to achieve airborne Design Assurance Level C. RSC system relies on machine-learning DNN technology which constitutes the focal point of our research and requires special certification considerations that we studied in [15] and implemented as a realistic example in this work. In order to meet the DAL C certification objectives for RSC, we utilize architectural mitigation along with ML-specific assurance techniques, following the DAL C ML workflow proposed in [15].

A. System Overview

Runway Sign Classifier (RSC) system is intended for the detection and classification of airport signs (Fig. 2). The system receives visual images from a forward-facing camera installed on the aircraft and uses a DNN for detection and classification tasks. In our baseline case study [13], we considered three variants of such a system depending on the level of automation and corresponding criticality levels:

- RSC-FB (Flight Bag). This variant serves as a pilot assistance system by providing information about detected runway signs to pilots e.g., via an enhanced vision system. Failures of such a system would have minor safety effects, as the pilot is still responsible for the continuous perception of the visual environment and retains full control of the aircraft. Consequently, the RSC-FB can be assigned the lowest design assurance level (DAL D).
- RSC-SM (Safety Monitor). This system variant provides information about detected airport signs to enable cockpit annunciation or automatic braking in hazardous taxing situations, such as crossing a holding position sign without the controller's approval. This capability provides a higher autonomy and criticality level and can be assigned DAL C.
- RSC-AU (Autonomy). This RSC variant is intended to support fully autonomous taxiing. In this context, the location and sign data directly contribute to the decision-making system responsible for autonomous aircraft control and preventing runway incursions. Given the potential safety impact, the system can be assigned DAL B or DAL A, depending on the aircraft category.

In our previous work [13], we implemented the first variant

Fig. 2. Airport sign

of the RSC-FB system and demonstrated its compliance with DAL D certification objectives. This study focuses on implementing the second variant, RSC-CM, using our custom ML workflow for DAL C [15]. In future work, we plan to address the third and most critical variant, RSC-AU.

B. System Architecture

A single-channel architecture of the original Runway Sign Classifier system presented in [13] is compatible with the DAL D custom ML workflow [12] but has to be redesigned to implement the architectural mitigation approach proposed DAL C ML-based systems in [15]. We upgraded the RSC system architecture by adding a second dissimilar DNN and a safety monitor component. Fig. 1 shows the implemented in this work DAL C RSC system architecture with two independent DNNs running in parallel; outputs of DNNs are continuously monitored by the safety monitor component and inhibited in case of divergence above the specified threshold. As justified in [15], the design assurance levels of the redundant dissimilar DNN components can be reduced to DAL D in such architecture while maintaining the DAL C for the overall system.

C. RSC Requirements

The functional and operational requirements that we developed for the DAL D implementation of the RSC system in our previous study [13] remain unchanged for the DAL C architecture upgrade, therefore we reused them in this case study. To address the DAL C workflow objectives for architectural mitigation measures [15], we added DAL C specific requirements for system components dissimilarity and a safety monitor as detailed in Table I. Furthermore, we elaborated the RSC operational conditions requirements into explicit requirements for RSC datasets (Table II) with the purpose of supporting data-specific verification activities discussed in Section II-D. For requirements authoring and management, we utilized the Polarion life-cycle management framework [16].

ID	Description
RSC-A1	The RSC system shall include two dissimilar DNN components independently performing sign detection function
RSC-A2	The RSC system shall include a safety monitor component that compares the outputs of the dissimilar DNN components. The safety monitor shall inhibit the system outputs if the outputs of the DNN components do not fall within the specified tolerance
RSC-A3	DNN components shall utilize dissimilar network architecture
RSC-A4	The RSC DNN components shall be trained and tested using independent datasets
RSC-A5	The RSC DNN components software shall be implemented using different programming languages
RSC-A6	The RSC DNN components shall be implemented using dissimilar electronic hardware
RSC-A7	The RSC DNN components shall be implemented by different individuals
RSC-A8	Verification of the RSC DNN components shall be performed by different individuals or groups

TABLE I
RSC SYSTEM ARCHITECTURE REQUIREMENTS

ID	Description
The RSC datasets shall contain sign images that have been captured within the premises of the following airports:	
KSFO	San Francisco International Airport
KBOS	Boston Logan International Airport
KSAN	San Diego International Airport
The RSC datasets shall contain sign images that have been captured under the following weather conditions:	
FAIR	Fair weather
RAIN	Rainy weather
SNOW	Snowy weather
FOG	Foggy weather
The RSC datasets shall contain sign images that have been captured during the following time of day:	
MRNG	Morning
DUSK	Dusk
AFTN	Afternoon
DAWN	Dawn
The RSC datasets shall contain sign images that have been captured within the following distance ranges:	
DS10	10 to 12 meters
DS12	12 to 14 meters
DS14	14 to 16 meters
The RSC datasets shall contain sign images captured from points at an elevation above ground level within the following ranges:	
EL10	1.0 to 1.3 meters
EL13	1.3 to 1.6 meters
EL16	1.6 to 1.9 meters
The RSC datasets shall contain sign images captured within the following ranges of the camera's lateral offset from the sign center line:	
LO00	0 to 0.7 meters
LO07	0.7 to 1.4 meters
LO14	1.4 to 2 meters

TABLE II
RSC DATA REQUIREMENTS

D. ML Data Management

In ML development workflows, datasets play a crucial role as they directly define the system behavior and therefore serve as high-level software requirements [13]. In this case study, we address the independence aspects of datasets used for the different DNN components to satisfy the dissimilarity requirements outlined in Section II-B. Additionally, we study the specific aspects of data configuration management and data verification that have to be addressed in a safety-critical context but are not covered well by the existing assurance practices.

1) Datasets Independence: As discussed in [15], a significant factor that can contribute to common errors in DNN components is the use of a shared dataset for training and testing across different DNNs. To address this aspect, we developed a separate dataset for the Faster R-CNN DNN component by utilizing the X-Plane[3] flight simulator to generate airport visual scenes. X-Plane is one of the most advanced flight simulators that offers exceptional capabilities in generating photo-realistic scenery and is also used for scenario-based pilot training [17]. For the YOLOv2 DNN component, we reused the dataset generated using the FlightGear flight simulator in the scope of our previous work [13]. To ensure data integrity

[3]https://www.x-plane.com/

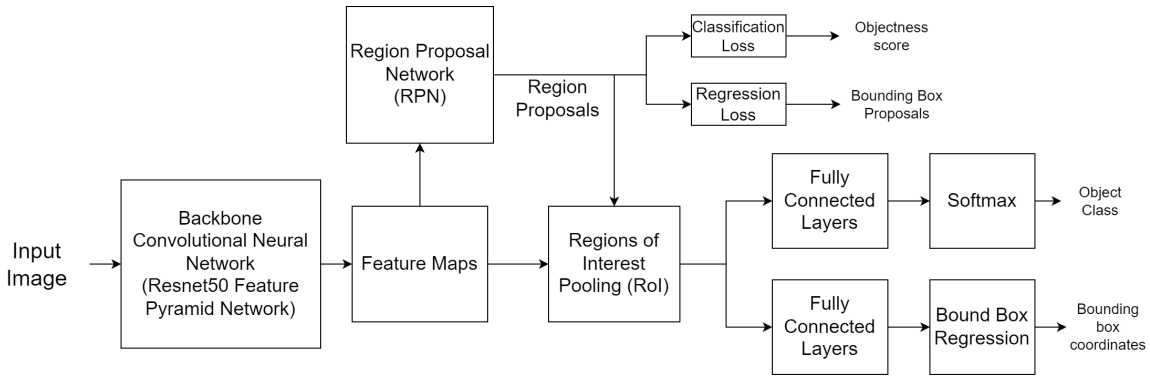

Fig. 3. Faster R-CNN Architecture

and prevent inadvertent data leakage, we implemented separate Git repositories for managing the FlightGear and X-Plane datasets. This segregation ensures that each dataset remains independent.

2) Data Configuration Management: ML datasets are typically characterized by a large number of elements, typically images, and videos in compressed binary format. Traditional configuration management systems such as Git[4] or SVN[5] can experience performance issues and other limitations when tracking the configuration of binary files [18]. To address these issues in the scope of our case study, we used the DVC[6] extension to the Git system to manage the configuration of our Runway Sign Classifier datasets. DVC is an open-source tool designed for machine learning and data-science applications. DVC enables storage of the binary data on a remote server allowing Git to manage a textual file with tracking reference of binary files.

E. System Implementation and Integration

In this work, we focus on the implementation of the dissimilar Python-based R-CNN DNN component, the safety monitor component, and their integration with the YOLOv2 DNN component implemented in our previous case study [13]. YOLOv2 (Figure 4) is a widely-used object detection network known for its speed and relatively small size [19]. We used the MATLAB implementation of the YOLOv2 DNN `yolov2ObjectDetector`[7] utilizing the DarkNet-19 backbone network and conducting the YOLOv2 training employing the capabilities of the MATLAB Computer Vision Toolbox[8].

The implementation and integration of the video camera and video pre-processing component are not included in this study because these components can be implemented and certified using traditional non-ML software and hardware assurance practices that fall outside the scope of this research.

[4]https://git-scm.com/
[5]https://subversion.apache.org/
[6]https://dvc.org/
[7]https://www.mathworks.com/help/vision/ref/yolov2objectdetector.html
[8]https://www.mathworks.com/help/vision/index.html

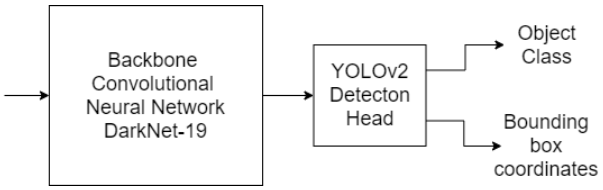

Fig. 4. YOLOv2 DNN Architecture

1) DNN components: The YOLOv2 DNN, implemented in our prior work [13], was reused for the first RSC DNN component. Taking into account the dissimilarity requirements for DNN components provided in Section II-B, we opted for a Python-based implementation of the Faster R-CNN [20] as a DNN architecture (Figure 3) for the second RSC component. The Faster R-CNN architecture employs a two-stage detection approach, incorporating a Region Proposal Network (RPN) to predict potential object locations before performing main network predictions and classification. The Faster R-CNN DNN architecture (depicted in Figure 3) presents notable differences compared to the YOLOv2 architecture (shown in Figure 4). These differences are evident in the backbone network architecture, with Faster R-CNN employing a ResNet50 convolutional network with 50 layers, while YOLOv2 uses a DarkNet-19 network with 19 layers. Moreover, the overall detection approach differs, with Faster R-CNN utilizing an advanced two-stage prediction pipeline, whereas YOLOv2 employs a single-stage pipeline optimized for speed.

We implemented the Faster R-CNN DNN component using the `fasterrcnn_resnet50_fpn` model from the `torchvision`[9] Python package. Utilizing the dissimilar architecture of DNN models, different programming languages and diverse training frameworks for independent DNN components mitigates the risk of common errors in independent system channels. Furthermore, to mitigate the risk of human error, we assigned the development of the Faster R-CNN DNN and YOLOv2 components to different individuals.

To match the interfaces of the YOLOv2 DNN component [13], we adjusted the input and output layers parameters of

[9]https://pytorch.org/vision/main/index.html

Faster R-CNN DNN. We kept the default values of the other DNN parameters, such as the number of RPN input features and loss function balancing parameters, because the required component performance was achieved during training without other hyperparameters tuning.

The Faster R-CNN training process utilized the independent X-Plane dataset described in Section II-D. To handle the training dataset, we employed the `torch.transforms`[10] and `CV2`[11] Python packages for various tasks, including the conversion of dataset images into PyTorch tensors to provide compatibility with the model training framework.

To determine the DNN detection thresholds, we evaluated the trained Faster R-CNN DNN using a precision/recall metric [21]. With a selected confidence and intersection threshold of 95% delivering the required average precision and highest possible recall of the model, the final Faster R-CNN DNN achieved an average detection precision of 100% on the test dataset.

2) Safety monitor component: The safety monitor component implements protection against DNN failures that can result in incorrect system output. The safety monitor compares the outputs (bounding box coordinates and sign class) of the dissimilar YOLOv2 and Faster R-CNN DNNs and passes through the output of the first DNN if it matches the outputs of the second DNN. If the different DNN components do not return the same sign class or the divergence between bounding box coordinates is above the threshold, the safety monitor inhibits the system output by setting the signal validity flag to FALSE. We selected the Intersection over Union (IoU) metric [22] for measuring the coherence of the bounding boxes coordinates between the different DNN components. The safety monitor IoU is calculated by dividing the area of intersection between the regions predicted by different DNN components by the area of their union (Figure 6).

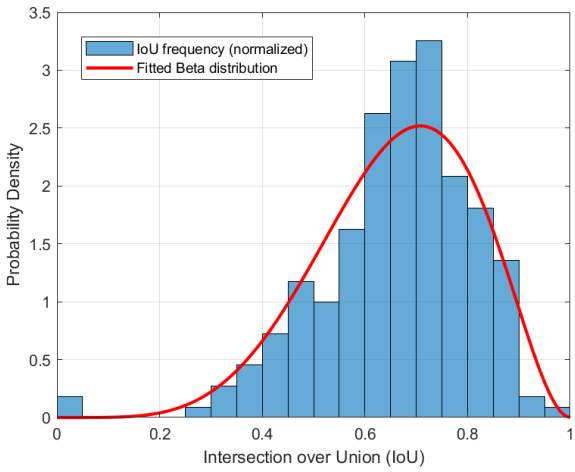

Fig. 5. Distribution of IoU for dissimilar DNN components

The IoU threshold for the safety monitor component was determined experimentally by analysis of the IoU distribution

[10]https://pytorch.org/vision/main/transforms.html
[11]https://pypi.org/project/opencv-python/

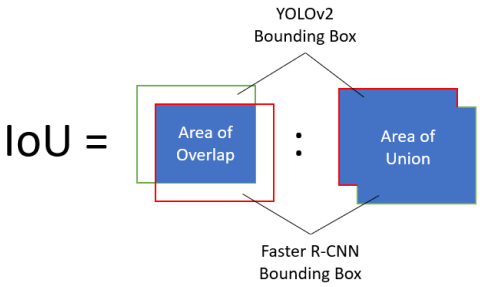

Fig. 6. Intersection over Union

over the available test dataset. The experimental frequencies of the IoU values fit the Beta distribution with a mean of 0.66 and a standard deviation of 0.54. Figure 5 shows the observed IoU value frequencies on the FlightGear testing dataset and the probability density of the fitted Beta distribution. To ensure the system availability above 95%, the safety monitor threshold for the IoU value was set to 0.32.

F. ML Specific Verification

Verification activities serve the purpose of identifying potential errors introduced during development. In this case study, we focus on the ML-specific data verification techniques which complement testing, reviews, and analysis methods that we studied in our previous work to fulfill the verification objectives required for DAL D systems [13].

1) Data Verification: ML datasets representing high-level software requirements in our workflow entail the application of typical objectives for requirements verification, such as verification of accuracy, consistency, and compliance with upstream requirements. However, ML datasets possess specific properties, such as higher complexity and size, that are not explicitly addressed in the current requirements verification practices outlined in DO-178C [23] and ARP-4754 [14]. In the scope of this case study, we implement a data traceability approach to address these specific properties of ML datasets. This approach is based on creating bi-directional links between textual data requirements and DNN datasets by tagging data files during the data preparation process.

To collect and analyze traceability information for datasets, we developed the DataTrace extension for the SimPol traceability management tool [24], originally created for model-based design workflows. DataTrace automates the fetching of requirements from the requirements management system Polarion [16] and creates and stores links to the linked datasets.

DataTrace also provides a capability to generate data traceability matrices (Fig. 7) and data requirements coverage charts (Fig. 8) from the collected data traceability information. We used the data traceability matrix to review by sampling the compliance of dataset elements with data requirements and analyze the traceability of dataset elements to requirements. The data requirement coverage chart (Fig. 8) was used to

RequirementID	Tag	FileSet	NumberOfItems	Files
"RSD-164"	"FAIR"	[228×1 string]	228	"Open Datastore"
"RSD-174"	"DUSK"	[144×1 string]	144	"Open Datastore"
"RSD-184"	"AGL"	[552×1 string]	552	"Open Datastore"
"RSD-162"	"KSAN"	[48×1 string]	48	"Open Datastore"
"RSD-172"	"MRNG"	[156×1 string]	156	"Open Datastore"
"RSD-160"	"KBOS"	[218×1 string]	218	"Open Datastore"
"RSD-170"	"FOG"	[96×1 string]	96	"Open Datastore"
"RSD-180"	"DIST"	[552×1 string]	552	"Open Datastore"
"RSD-168"	"SNOW"	[48×1 string]	48	"Open Datastore"
"RSD-178"	"DAWN"	[180×1 string]	180	"Open Datastore"
"RSD-166"	"RAIN"	[108×1 string]	108	"Open Datastore"
"RSD-176"	"AFTN"	[72×1 string]	72	"Open Datastore"

Fig. 7. Data Traceability Matrix

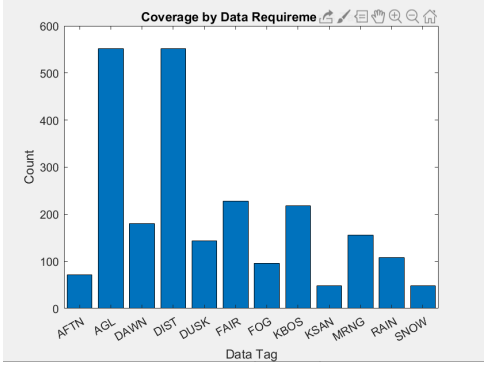

Fig. 8. Data Requirements Coverage Chart

analyze the coverage of data requirements, including the characteristics of coverage distribution.

III. Certification Compliance Analysis

In this section, we elaborate on the analysis of DO-178C DAL C objectives provided in [15] by mapping the objectives to the actual activities performed and artifacts created in the scope of DAL C RSC development. Table III includes the list of DAL C objectives, their applicability by the DO-178C software levels and the compliance analysis column. This table complements the DAL D objectives analysis presented in [13] and therefore doesn't include DAL D objectives which are also applicable to DAL C. Since the generation of target source code from DNNs has no significant disparities compared to traditional software assurance practices covered by DO-178C, the source code development and verification objectives are not included in the analysis. Also, this analysis does not encompass the objectives related to planning, configuration management, quality assurance, and certification liaison processes. These processes are not significantly affected by ML-specific challenges and can be fulfilled using existing assurance practices [15]. Overall 14 out of 23 DO-178C and DO-330 objectives included in the analyzed DAL C subset have been directly accomplished, 4 objectives for executable code testing have been covered at the RSC DNN model level, and 5 objectives have been justified through the application of architectural mitigation.

IV. Conclusions and Future Work

This paper focuses on the certification and safety aspects of machine-learning systems used in safety-critical airborne applications. To demonstrate the approach for addressing the certification issues of such systems, we conducted a case study of an onboard system utilizing deep neural networks for the detection and classification of airport runway signs. We demonstrated that machine learning systems can attain Design Assurance Level C by utilizing our custom development workflow in combination with an architectural mitigation approach and selected novel ML-specific data verification methods. Building upon our previous work that studied the Design Assurance Level D certification workflow, this case study involves the use of two dissimilar DNN components running in parallel. As part of the architectural mitigation approach, we integrated the safety monitor component, which continuously compares the outputs of dissimilar DNNs to detect potential errors in DNN outputs and prevent error propagation.

Dissimilarity of independent DNNs is achieved by using dissimilar network architectures, training frameworks, and data sets. On a more implementation-oriented level, we utilized different programming languages and having different team members doing the implementation. The distribution of detection divergence for dissimilar DNNs measured on the available dataset fits the normal distribution that correlates with the dissimilarity applied concepts. Based on the determined characteristics of the divergence distribution, we selected an optimal filtering threshold for the safety monitor component to ensure a target system availability of 95%.

To verify compliance of the implemented system with Design Assurance Level C, we conducted a thorough analysis and

justification of the DAL C objectives outlined in the DO-178C and DO-331 standards. Additionally, we mapped the analyzed objectives to the corresponding requirements specified in the latest ML-specific certification guidance published by EASA [25].

In our future work, we will further study novel verification techniques specific to ML technologies with respect to certification objectives. In particular, we plan to focus on statistical data quality, model stability and robustness analysis methods, and out-of-distribution detection techniques. Additionally, we aim to explore the aspects of ML models' dissimilarity with respect to the Run Time Assurance approach described in ASTM 3269 [26]. Lastly, we intend to conduct the qualification of ML data traceability tools to streamline manual activities of data verification following the approach proposed in [27].

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

REFERENCES

[1] "Concepts of design assurance for neural networks (CoDANN)," European Aviation Safety Agency, Tech. Rep., 2020.

[2] "Report. concepts of design assurance for neural networks (CoDANN) II," European Aviation Safety Agency, Tech. Rep., 2021.

[3] "Artificial intelligence roadmap 2.0. human-centric approach to AI in aviation," European Aviation Safety Agency, Tech. Rep., 2023.

[4] "Artificial intelligence in aeronautical systems. statement of concerns." EUROCAE, Tech. Rep. AIR6988, 2021.

[5] F. Kaakai, K. Dmitriev, S. Adibhatla, E. Baskaya, and et al., "Toward a Machine Learning Development Lifecycle for Product Certification and Approval in Aviation," *SAE Int. J. Aerosp. 15(2):2022*, 2022.

[6] "Neural network based runway landing guidance for general aviation autoland," Federal Aviation Administration, Tech. Rep. DOT/FAA/TC-21/48, 2022.

[7] H. Delseny, C. Gabreau, A. Gauffriau, B. Beaudouin, L. Ponsolle, L. Alecu, H. Bonnin, B. Beltran, D. Duchel, J.-B. Ginestet *et al.*, "White paper machine learning in certified systems," *arXiv preprint arXiv:2103.10529*, 2021.

[8] C. Torens, F. Juenger, S. Schirmer, S. Schopferer, D. Zhukov, and J. C. Dauer, "Ensuring Safety of Machine Learning Components Using Operational Design Domain," in *AIAA SCITECH 2023 Forum*, 2023, p. 1124.

[9] N. Escudero, P. Costas, M. W. Hardt, and G. Inalhan, "Machine Learning Based Visual Navigation System Architecture for Aam Operations with A Discussion on its Certifiability," in *2022 Integrated Communication, Navigation and Surveillance Conference (ICNS)*, 2022, pp. 1–15.

[10] M. S. Feather, S. Guerrini, P. C. Slingerland, and M. Spolaor, "Assurance Guidance for Space Mission use of Data-Driven Machine Learning," in *2023 IEEE Aerospace Conference*. IEEE, 2023, pp. 1–10.

[11] D. B. Abeywickrama, J. Wilson, S. Lee, G. Chance, P. D. Winter, A. Manzini, I. Habli, S. Windsor, S. Hauert, and K. Eder, "AERoS: Assurance of Emergent Behaviour in Autonomous Robotic Swarms," *arXiv e-prints*, pp. arXiv–2302, 2023.

[12] K. Dmitriev, J. Schumann, and F. Holzapfel, "Toward certification of machine-learning systems for low criticality airborne applications," in *2021 AIAA/IEEE 40th Digital Avionics Systems Conference (DASC)*. IEEE, 2021, pp. 1–10.

[13] ——, "Toward Design Assurance of Machine-Learning Airborne Systems," in *AIAA SciTech 2022 Forum*, 2022, p. 1134.

[14] *Guidelines for Development of Civil Aircraft and Systems*, SAE International Std. SAE ARP4754A, 2010.

[15] K. Dmitriev, J. Schumann, and F. Holzapfel, "Towards Design Assurance Level C for Machine-Learning Airborne Applications," in *2022 IEEE/AIAA 41st Digital Avionics Systems Conference (DASC)*. IEEE, 2022, pp. 1–6.

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

[17] B. Williams, *Scenario-Based Training with X-Plane and Microsoft Flight Simulator: Using PC-Based Flight Simulations Based on FAA-Industry Training Standards*. John Wiley & Sons, 2011.

[18] P. Janardhanan, "Project repositories for machine learning with TensorFlow," *Procedia Computer Science*, vol. 171, pp. 188–196, 2020, third International Conference on Computing and Network Communications (CoCoNet'19). [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050920309856

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

[20] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *Advances in neural information processing systems*, vol. 28, 2015.

[21] T. Saito and M. Rehmsmeier, "The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets," *PloS one*, vol. 10, no. 3, p. e0118432, 2015.

[22] H. Rezatofighi, N. Tsoi, J. Gwak, A. Sadeghian, I. Reid, and S. Savarese, "Generalized intersection over union: A metric and a loss for bounding box regression," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 658–666.

[23] *Software Considerations in Airborne Systems and Equipment Certification*, RTCA, Inc. Std. RTCA DO-178C, 2011.

[24] M. T. Hochstrasser, "Modular model-based development of safety-critical flight control software," Ph.D. dissertation, Technische Universität München, 2020.

[25] "EASA concept paper: First usable guidance for level 1&2 machine learning applications," European Aviation Safety Agency, Tech. Rep., 2023.

[26] *Standard Practice for Methods to Safely Bound Behavior of Aircraft Systems Containing Complex Functions Using Run-Time Assurance*, ASTM International Std. F3269, 2021.

[27] K. Dmitriev, F. Kaakai, M. Ibrahim, U. Durak, B. Potter, and F. Holzapfel, "Tool Qualification Aspects in ML-Based Airborne Systems Development," in *Software Engineering 2023 Workshops*. Gesellschaft für Informatik eV, 2023.

[28] *Road vehicles — Functional safety — Part 6: Product development at the software level*, International Organization for Standardization Std. ISO 26 262-6, 2018.

TABLE III
DAL C OBJECTIVES ANALYSIS

DO-178C/DO-331 Objectives		Software Levels A	B	C	D	E	Compliance Analysis
A-2#4	Low-level requirements are developed.	x	x	x			Trained RSC DNN models represent software design (low-level requirements and architecture); the objective is satisfied. This corresponds to the objective IMP-03 of the EASA Guidance [25].
A-2#5	Derived low-level requirements are defined and provided to the system processes, including the safety assessment process.	x	x	x			The *traceability issue* discussed in [15] makes it practically infeasible to directly identify derived requirements associated with RSC DNNs. The EASA Guidance [25] does not have corresponding objectives. The justification for this objective relies on the implemented architectural mitigation, which includes utilizing two dissimilar RSC DNNs and a safety monitor.
A-3#4	High-level requirements are verifiable.	x	x	x			The RSC system, component and data requirements written in textual form have been peer-reviewed to accomplish this objective. This corresponds to the objective DA-04 of the EASA Guidance [25].
A-3#5	High-level requirements conform to standards.	x	x	x			Same as for the objective A-3#4.
A-3#7	Algorithms are accurate.	i	i	x			Same as for the objective A-3#4.
A-4#1	Low-level requirements comply with high-level requirements.	i	i	x			This objective is achieved through the testing of the RSC DNNs, which represent low-level software requirements in the implemented custom workflow. This corresponds to the combination of the objectives LM-09 and LM-10 of the EASA Guidance [25].
A-4#2	Low-level requirements are accurate and consistent.	i	i	x			Same as for the objective A-4#1.
A-4#5	Low-level requirements conform to standards.	x	x	x			This objective has been fulfilled by conducting peer reviews of the RSC DNN models representing software design. The EASA Guidance [25] has no corresponding objectives; this could indicate a potential gap in the provided guidance.
A-4#6	Low-level requirements are traceable to high-level requirements.	x	x	x			Similar to objective A-2#5, this objective is impacted by the *traceability issue*. The EASA Guidance [25] does not include corresponding objectives. The implemented architectural mitigation measures support the justification for this objective.
A-4#7	Algorithms are accurate.	i	i	x			Same as for the objective A-4#1.
A-4#8	Software architecture is compatible with high-level requirements.	i	x	x			Same as for the objective A-4#1.
A-4#9	Software architecture is consistent.	i	x	x			Same as for the objective A-4#1.
A-4#12	Software architecture conforms to standards.	x	x	x			Same as for the objective A-4#5.
MB.A-4#MB14	Simulation cases are correct	i	x	x			Since DNN model testing is used to fulfill the objectives for low-level requirements verification; it is necessary to verify the correctness of the DNN model test cases to ensure comprehensive testing. This objective is achieved through verification of requirements coverage by test data set as outlined in Section II-F. This is partially addressed (for test dataset only) by the objective DM-14 the EASA Guidance [25].
MB.A-4#MB15	Simulation procedures are correct	i	x	x			Similar to the rationale behind objective MB.A-4#MB14, it is necessary to verify the correctness of test procedures for the DNN model. RSC DNNs test procedures have been peer-reviewed to satisfy this objective. The EASA Guidance [25] has no corresponding objectives; this could indicate a potential gap in the provided guidance.
MB.A-4#MB16	Simulation results are correct and discrepancies explained	i	x	x			Similar to the rationale behind objective MB.A-4#MB14, it is necessary to verify DNN testing results. These testing results have been peer-reviewed to satisfy this objective. This is implicitly addressed by the objective LM-10 of the EASA Guidance [25].
A-6#3	Executable Object Code complies with low-level requirements.	i	i	x			To fulfill this objective, back-to-back testing of the target executable object code against the RSC DNN model can be utilized as described in [28]. Since the deployment of RSC DNN on target hardware was determined to have no significant disparities compared to traditional software assurance practices covered by DO-178C [15], the testing objectives were implemented at the RSC DNN model level only, see A-4 and MB.A-4 objectives. This is covered by the objective IMP-03 of the EASA Guidance [25].
A-6#4	Executable Object Code is robust with low-level requirements.	i	x	x			Same as for the objective A-6#3.
A-7#1	Test procedures are correct.	i	x	x			Verification of test procedures through reviews can be utilized to satisfy this objective. This activity was replaced by MB.A-4#MB15 in the scope of this case study, see objective A-6#3.
A-7#2	Test results are correct and discrepancies explained.	i	x	x			Verification of test results through reviews can be utilized to satisfy this objective. This activity was replaced by MB.A-4#MB16 in the scope of this case study, see objective A-6#3.
A-7#4	Test coverage of low-level requirements is achieved.	i	x	x			Due to the inherent *coverage issue* discussed in [15], there are no relevant metrics for the RSC DNN coverage analysis. The EASA Guidance [25] has no corresponding objectives. The justification for this objective is claimed from the implemented architectural mitigation, which involves the utilization of two dissimilar RSC DNNs and a safety monitor.
A-7#7	Test coverage of software structure (statement coverage) is achieved.	i	i	x			As discussed in [15], structural coverage metrics at the DNN source code level are not representative due to the inherent *coverage issue* and have not been included in the scope of this work. The EASA Guidance [25] has no corresponding objectives. The justification for this objective is claimed from the implemented RSC architectural mitigation.
A-7#8	Test coverage of software structure (data coupling and control coupling) is achieved.	i	i	x			Analysis of data and control coupling can be implemented using traditional methods to satisfy this objective. This is covered by the objective IMP-03 of the EASA Guidance [25]. The verification activities at the source code level have not been included in the scope of this case study, see the objective A-6#2.