Safety-by-Design Methodology for Simulation-Based Dataset Creation and AI Model Development

Applying Selected Objectives of the *EASA Guidance for Level 1 & 2 Machine Learning Applications* to a Runway Object Detection Use Case

Master's Thesis

Author: Mina Randolf (Student ID: 7354702)
Supervisor: Prof. Dr. Markus Weinmann
Co-Supervisor: Christopher Martin Coors

Faculty of Management, Economics and Social Sciences
University of Cologne

June 3, 2026

Eidesstattliche Versicherung

Hiermit versichere ich an Eides statt, dass ich die vorliegende Arbeit selbstständig und ohne die Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten und nicht veröffentlichten Schriften entnommen wurden, sind als solche kenntlich gemacht. Die Arbeit ist in gleicher oder ähnlicher Form oder auszugsweise im Rahmen einer anderen Prüfung noch nicht vorgelegt worden. Ich versichere, dass die eingereichte elektronische Fassung der eingereichten Druckfassung vollständig entspricht.

Die Strafbarkeit einer falschen eidesstattlichen Versicherung ist mir bekannt, namentlich die Strafandrohung gemäß § 156 StGB bis zu drei Jahren Freiheitsstrafe oder Geldstrafe bei vorsätzlicher Begehung der Tat bzw. gemäß § 161 Abs. 1 StGB bis zu einem Jahr Freiheitsstrafe oder Geldstrafe bei fahrlässiger Begehung.

Mina Randolf

Köln, den 02.06.2026

Abstract

Foreign Object Debris (FOD), referring to unwanted objects or materials on airport runways, poses a significant safety risk to flight operations, causing billions of dollars' worth of damage to the aviation industry each year. Artificial intelligence offers a promising solution for the automated and early detection of such objects. However, the use of machine learning in safety-critical aviation applications requires a structured development process that meets regulatory requirements. In its concept paper *Guidance for Level 1 & 2 Machine Learning Applications*, the European Union Aviation Safety Agency (EASA) has presented a set of objectives. These objectives remain at an abstract level and provide little guidance on how they should be implemented in practice. This results in a gap between regulatory guidance and concrete development practice.

This thesis addresses the gap by developing a Safety-by-Design methodology that integrates six selected EASA objectives. The methodology covers the Data Management, Learning Process Management, and Model Training phases of a structured development process. It involves the ODD-to-Data Traceability Matrix, the operationalization of data quality requirements, controlled data processing, and independent dataset partitioning. The methodology is supplemented by the systematic definition of the model architecture, the learning process, the training environment, and the execution and documentation of training.

The developed methodology was validated using the specific use case of object detection on Runway 23 at Hamburg Airport during the approach. This dataset consists of 13 496 annotated frames from 56 simulation runs across eight object classes. It was created in Unreal Engine 5.3 using ProjectAirSim. A YOLO11m model was fine-tuned on this dataset in a two-stage process, achieving a detection performance of mAP@0.5 = 0.263 after full fine-tuning This represents a 27.8% relative improvement over head-only training and more than two orders of magnitude above the zero-shot baseline of 0.0008.

These results demonstrate that the developed Safety-by-Design methodology successfully translates the selected EASA objectives into a practical, applicable development process, ensuring regulatory compliance and traceability while providing a structured foundation that measurably improves model performance through systematic data quality assurance and controlled training. This thesis bridges the gap between abstract regulatory guidance and concrete development practice, providing a replicable and validated foundation for future research on verification, integration, and certification of AI-based aviation systems.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Motivation and Problem Context

Aviation is considered one of the safest modes of transportation, yet industry analyses consistently show that more than half of all fatal accidents occur during takeoff and landing. This highlights that a disproportionate number of incidents happen when aircraft operate close to the ground, with little margin for error [2].

A persistent cause for severe incidents is foreign object debris (FOD) on the runway. FOD refers to unwanted objects in safety-critical areas that can cause damage to aircraft, equipment, or personnel [3]. During critical phases like takeoff and landing, even a small object can lead to disastrous consequences. An example of such an incident is the crash of the Concorde on Air France Flight 4590, which was caused by small pieces of metal on the runway. The debris ruptured a tire, punctured a fuel tank, and caused a fatal fire, resulting in the loss of 113 lives, including passengers and ground crew [4]. This incident demonstrates how dangerous even small objects on the runway can be. Runway safety incidents are not limited to small debris. A recent accident, the collision between an Air Canada aircraft and a fire truck at LaGuardia Airport in March 2026, in which both pilots were killed [5], demonstrate that larger objects and vehicles on the runway pose an equally serious threat. Additionally, unauthorized drone activity near runways poses another growing security concern, as collisions can cause severe engine damage [3]. These examples illustrate that FOD can range from small debris to vehicles or unauthorized drones, underscoring the need for comprehensive automated detection systems.

Beyond catastrophic events, FOD leads to significant financial damage. Boeing has estimated that FOD costs the aviation industry about $4 billion per year [6]. When resulting indirect costs such as flight delays, cancellations, and schedule disruptions are included, the annual global cost of FOD is estimated to be $22.7 billion per year [7]. The world's 300 largest airports collectively report over 10,000 FOD-related incidents annually [8].

To mitigate the mentioned risks, runways are traditionally inspected through visual checks by personnel. This manual approach is labor-intensive, which can limit the frequency with which inspections are performed [3]. Given the scale of modern airport operations, automated FOD detection systems have become a key priority for enhancing safety and reducing costly damage and delays.

Artificial Intelligence (AI) offers a particularly promising approach to this challenge, as AI-based object detection systems can continuously monitor runway surfaces and identify hazards earlier and more reliably than manual inspection alone [3, 9]. Furthermore, the complexity of FOD detection, where objects vary

widely in size, shape, and appearance, makes AI-based visual systems a particularly promising approach compared to manual inspection. Additionally, many airports and aircraft already have camera-based infrastructure installed, which could be utilized for AI-based FOD detection [3]. Rule-based alternatives are limited in their ability to achieve a comparable performance across such a diverse range of hazards [3]. Machine-learning-based visual systems have opened promising avenues for addressing the FOD problem by automatically detecting hazards earlier than human operators alone [3, 9].

AI-based detection systems offer a promising opportunity to improve runway safety and reduce the financial burden caused by FOD-related incidents. Nevertheless, the aviation industry faces the challenge that it is subject to strict certification processes due to its many safety-critical applications [1, 10]. Aviation is expected to experience a considerable overall surge in AI-based applications, with experts projecting a growth rate of 35 % per year [11]. Applying AI in aviation therefore urgently requires a structured development approach that addresses the strict regulatory and safety-critical characteristics of this industry. This thesis therefore develops a structured Safety-by-Design methodology and validates it in practice through the creation of a simulation-based dataset and the training of a prototype detection model for FOD detection."

1.2 Research Gap

In recent years, the aviation industry has made significant progress in integrating AI into safety-critical applications [12]. It has become apparent that integrating machine learning (ML) into aviation introduces new challenges when needing to ensure system safety. Unlike classical, rule-based software systems, ML models depend heavily on data-driven processes, which hide how the system makes decisions and complicates continuous verification during development [10, 13, 14].

In response, the European Union Aviation Safety Agency (EASA) published multiple reports regarding the use of AI in aviation. One of those reports, *EASA Concept Paper: Guidance for Level 1 & 2 machine learning applications [1]*, is of particular importance for this thesis. It defines guidelines for the safe development of AI-based systems for aviation applications. The concept paper distinguishes three levels of AI applications based on the degree of authority given to the system. Level 1 covers AI systems that assist human operators, who retain full decision-making authority. Level 2 introduces human-AI teaming, in which the system can autonomously implement actions under continuous human oversight [1]. However, it raises the question of how these requirements can be practically implemented, especially in a safety-critical use case. The lack of a

structured, practical development methodology makes it difficult for developers to build certifiable ML solutions.

This thesis aims to address this gap by providing, to the best of the author's knowledge, the first structured Safety-by-Design methodology that translates selected EASA objectives into concrete development steps, validated through the creation of a synthetic dataset and the training of a prototype object detection model for runway FOD detection.

1.3 Research Question

To address the identified gap between regulatory guidance and practical implementation, this thesis focuses on designing a Safety-by-Design methodology that translates selected objectives from the *EASA Concept Paper: Guidance for Level 1 & 2 Machine Learning Applications* into a concrete and implementable development approach.

Although these objectives provide a comprehensive basis for certifying AI-based systems in aviation, the scope of a Master's Thesis does not allow for the examination of the entire range of objectives. This work, therefore, focuses on a manageable and meaningful selection. Six objectives were chosen because they directly shape the development of safe, reproducible, and well-documented machine learning components, as listed in Table 7 in the Appendix.

By structuring the methodology around these objectives, this thesis ensures that the development process follows a step-by-step implementation of EASA's Safety-by-Design principles rather than remaining a purely technical exercise.

The overarching research question guiding this thesis is therefore formulated as follows: *What should a Safety-by-Design methodology, in line with selected objectives of the EASA Guidance for Level 1 & 2 Machine Learning Applications, look like to develop a simulation-based dataset and a prototype AI model for the detection and classification of safety-critical runway objects (FOD)?*

1.4 Thesis Structure

This master's thesis is structured into six chapters. The current chapter provides an introduction to the topic and the research question this thesis aims to answer. The remainder is organized as follows. Section 2 reviews the existing literature across two research streams. The first covers safety and certification frameworks for AI-based aviation systems, focusing on regulatory guidance and the ODD concept. The second stream addresses object detection for runway safety, including sensing technologies and deep learning–based FOD detection methods. The chapter concludes by identifying research gaps and positioning the contribution of

this thesis. Section 3 describes the methodology in detail. Based on the selected objectives from the *EASA Concept Paper: Guidance for Level 1 & 2 Machine Learning Applications*, a structured Safety-by-Design methodology for simulation-based dataset creation and AI model development is proposed. The methodology translates the corresponding regulatory objectives into concrete development steps for AI-based runway object detection systems. Section 4 practically applies the methodology from Section 3 to a runway object detection use case. A synthetic dataset is generated using a flight simulator, followed by the adaptation, development, and training of a machine learning model. The experimental setup is described in detail, and each step is linked to the corresponding EASA objectives to demonstrate alignment between theory and implementation. In Section 5, the results of the implemented use case are presented. This includes the training, validation, and testing performance of the machine learning model, and the assessment of the methodology with regard to the selected EASA objectives. Section 6 summarizes the main findings, discusses limitations, and outlines directions for future research. It also reflects on the contribution and implications for theory and practice. The thesis concludes with perspectives on how this Safety-by-Design methodology contributes to advancing safe AI adoption in aviation, providing both a functional proof of concept and a replicable case study for future development efforts.

2 Related Work

This chapter reviews the existing literature related to the developed methodology and applied use case of this thesis. The review is organized into two research streams. The first stream, Section 2.1, covers safety and certification frameworks for AI-based systems in aviation, and the second stream, Section 2.2, addresses object detection for runway safety. Section 2.3 identifies gaps in the existing literature and positions the contribution of this thesis.

2.1 Safety in AI-Based Aviation Systems

2.1.1 Certification and Assurance Frameworks for Aviation AI

Safety-critical aviation systems have long been developed following established certification standards, including ED-12C and DO-178C for software and ARP-4754A for system development [15, 16, 17]. These standards require precise and fully specifiable system behavior, where each requirement traces back to a specific implementation [10, 14, 18].

Machine learning components challenge this concept, as their behavior is learned from data rather than explicitly programmed and therefore cannot be fully predicted for unknown inputs [10, 19]. This is why traditional certification standards are unable to ensure the safety of data-driven machine learning systems [1, 20, 21]. The SAE Committee on AI in aviation has identified this limitation as a key challenge for the industry [22].

In response to this, EASA has developed a framework for the safe development of AI-based systems, from the initial concept paper for level 1 applications [23] to the AI Roadmap 2.0 [24] to the current concept paper for level 1 & 2 machine learning applications [1]. The paper introduces the integrated learning assurance process and proposes a W-shaped development process as the central process model. This framework extends the classic V-model of software development with a data-driven component [23]. For example, Durand et al. explore practical elements for machine learning certification [25], SAE AIR6988 formulates fundamental requirements from an industry perspective [22], and more recent work developed overarching engineering frameworks for specific certification scenarios [12, 26]. These approaches all emphasize transparency, traceability, and reproducibility as fundamental characteristics for trustworthy machine learning systems [24].

2.1.2 Operational Design Domain and Data-Centric Safety

The concept of the operational design domain (ODD) originated in the automotive sector [20, 27]. It was introduced to describe the specific operating conditions

under which a driver assistance system is designed to operate safely [28, 29]. Since then, this concept has been applied to other domains, including aviation [27, 30]. In general, for autonomous systems, a cross-domain ODD definition process has been proposed to structure the initialization and step-by-step refinement of ODD attributes [29, 31, 32].

In aviation, the EASA has adopted and further developed the ODD concept for AI-based systems. An important feature of this approach is differentiating between the system-wide Operational Domain (OD) and the AI/ML Constituent ODD. While the ODD describes the overall system's general operating conditions, the AI/ML Constituent ODD defines the ML component's specific operating conditions and simultaneously serves as a data management framework [1, 31, 32]. This positions the ODD as a data management framework that directly links operational conditions to data collection, processing, and representativeness requirements [21, 32, 33]. Several studies have developed methods to structure the transition from the broader system ODD to the data-centered AI/ML constituent ODD [30, 34, 35]. Christensen et al. emphasize the need to translate ODD parameters systematically into data requirements to establish a traceable link between operating conditions and training data [12].

A closely related question is how to collect ODD-compliant data in practice. For many safety-critical applications, collecting real-world data under controlled conditions is difficult or impossible [10, 36]. It has become an important approach for covering the operating conditions defined in the ODD [19, 29, 35]. Gupta et al. demonstrate how to generate synthetic data for AI-based aviation systems based on operational scenarios [36]. Rüter et al. highlight how simulations can be used to address gaps in ODD coverage that cannot be filled with real-world data [37]. One well-known limitation of this approach is the simulation-reality gap, or the missing transparency between synthetic and real data [13, 36]. This discrepancy can limit the transferability of trained models [10, 13, 38]. Dieter et al. quantify this gap in the field of drone-based detection and show that it can be minimized with a targeted data strategy [38]. A further advantage is automated annotation. Since every object's position is known in simulation, bounding boxes are calculated automatically, eliminating labeling effort and uncertainty [39, 40, 41]. For the runway use case in this study, simulation-based data generation is advantageous and, due to the lack of real-world datasets, is the only applicable option [3, 13, 41, 42].

2.2 Object Detection for Runway Safety

2.2.1 FOD Detection: Problem and Sensing Approaches

The literature has examined various techniques for the automated detection of foreign objects on runways. Automated monitoring is increasing and regarded as essential, given that traditional methods and visual observation by air traffic controllers are frequently inadequate for the complex and extensive environments of modern airports [43]. One possible solution is a sensor-based approach. Radar-based systems are widely used because they can detect stationary objects regardless of weather conditions or time of day [44]. However, these systems are expensive to install and often require multiple antennas to provide comprehensive coverage [44]. Although lidar-based approaches offer high visual resolution, their performance can be affected by atmospheric conditions [45]. This is why camera-based systems have emerged as a promising alternative. They provide a high visual resolution while being considered a cost-effective solution for monitoring airport surfaces [3, 44]. In the context of deep learning methods for object detection, camera-based systems are very compatible. They enable continuous, scalable monitoring, making it an important advantage [3, 9, 43]. Existing camera-based FOD detection systems are mainly designed as ground-based systems that place cameras along the runway [3, 43, 45]. The use case considered in this work represents a different approach that uses a camera perspective from the aircraft itself during the landing approach. This perspective offers an additional detection window during the final approach. This is a critical moment, because the runway must be clear and does not require any additional ground-based infrastructure. There is little prior research addressing this specific perspective. The most relevant example is the LARD dataset, which covers the visual approach perspective, but focuses on recognizing runways rather than detecting objects [46, 47].

2.2.2 Deep Learning Architectures for Object Detection

Automatic object detection in image data is a central area of research in deep learning, increasing in popularity due to its ability to classify and localize objects within an image or video [48, 49]. Modern object detection architecture can be categorized into two basic categories, two-stage and one-stage detectors [49, 50]. Two-stage approaches like Faster R-CNN first generate potential regions and then classify them [9, 50]. While this kind of approach achieves high accuracy, the two-step process results in slower inference, making it less suitable for real-time applications [51]. Transformer-based architectures, like DEtection TRansformer (DETR), eliminate manual components such as anchor boxes and model object relationships [52]. However, they require large amounts of data and long training

times [52, 53]. On the other side, single-stage detectors, led by the YOLO (You Only Look Once) model family, combine classification and localization in a single process [54]. They offer significantly faster inference with comparable detection accuracy and have therefore become the preferred architecture for time-critical applications [49, 51, 55].

In the context of FOD and object detection at airports and especially on runways, the YOLO family has established itself as the leading approach in literature [3, 54]. Early work demonstrated that single-stage detectors are suitable for runway image data [56]. Other studies have evaluated different YOLO variants in airport environments, demonstrating their ability to perform competitively in real-world conditions [3, 9]. Especially important for deep learning-based FOD detection on runways is a suitable architectural design that can adapt to different lighting conditions and object sizes [53]. Further improvements were achieved by integrating attention mechanisms into YOLOv5, which enhanced the detection of small objects within railway and airport environments [57]. The architecture was most recently further developed with YOLO11 through improved feature extraction and more efficient network blocks [51, 58].

A common challenge that many of these studies face is the lack of large enough annotated real-world datasets for runway object detection applications [3, 9, 42]. Therefore, transfer learning has become a standard strategy [9, 59]. In this process, pre-trained models are fine-tuned for new tasks. Using weights pre-trained on large datasets such as COCO (Common Objects in Context) enables stable detection performance to be achieved even with limited domain-specific training data [59, 60, 61]. Lenhard et al. demonstrate in several papers that a two-phase training strategy, with first training only the detection head and then the entire network, reduces training instability with small datasets and improves overall performance [59, 61].

2.2.3 FOD Detection with Deep Learning and the Data Challenge

Although there are proven capabilities of deep learning architectures for object recognition, their practical application to FOD detection is limited. There is a lack of suitable training data. Publicly available, real-world annotated image datasets of runway objects from the perspective of a landing aircraft do not exist [3, 9]. Collecting such data would require extensive coordination with airport authorities as well as significant security measures [37, 41]. This data shortage is not just a random research gap but indicates a structural problem in the field of application.

In response, simulation-based data generation has become a key research approach. It is easy to set and adjust environmental parameters such as lighting or weather. Additionally, technical settings like the camera position can be precisely

controlled. Annotating the data for supervised learning would normally take up a lot of resources and is, manually, very time-consuming. With a synthetic dataset, labels can be generated automatically. And the amount of data can be scaled up as required [39, 40, 59, 62]. However, the literature consequently highlights the simulation-reality gap, as mentioned above [37, 38]. For related tasks such as drone-based detection, targeted data strategies, and domain-adapted training methods have been shown to reduce this gap [38, 59].

To date, only a few specific datasets are available for runway use case [3, 62]. While LARD (Landing Approach Runway Dataset) addresses visual runway recognition during the approach, it focuses on recognizing the runway itself rather than objects on it [46, 47, 62]. This is a notably different use-case [3]. An ODD-compliant, annotated dataset for detecting safety-critical objects on the runway from a landing approach perspective does not exist yet [62]. This work aims to address this issue by developing a synthetic dataset.

2.3 Contribution

The reviewed literature shows that foundations have been developed in the two relevant fields of research. There are regulatory frameworks for AI in aviation and deep learning-based object recognition, but an gap has emerged that this work addresses. In terms of regulation, the EASA Concept Paper presents objectives for the safe development of ML systems in aviation [1]. Existing work has addressed aspects of these objectives, like formulating the ODD definition process [31], developing engineering frameworks [12, 26], or extracting data requirements from the ODD [33]. A practical, end-to-end methodology that translates these objectives into concrete development steps and validates them against a real-world use case is still missing. In terms of applications, the literature on FOD detection shows that deep learning-based methods are effective for identifying objects in runway scenarios [3, 9, 56]. Neither real-world datasets nor ODD-compliant synthetic datasets exist for the specific use case of camera-based FOD detection from a landing aircraft's approach perspective [3, 9, 62]. Although simulation-based approaches to data generation have been tested for related tasks [32, 36, 39, 41], they have not yet been applied to a development process structured according to EASA objectives. This thesis addresses both gaps. It develops a Safety-by-Design methodology that translates selected European Aviation Safety Agency objectives into structured development steps. This methodology is validated by creating the synthetic dataset and training a YOLO11m model for runway object detection. By doing so, this work helps to close the gap between regulatory guidance and the practical implementation of certifiable AI systems in aviation.

3 Methodology Development

The methodology developed in this thesis translates selected objectives from the *EASA Concept Paper: Guidance for Level 1 & 2 Machine Learning Applications* [1] into concrete development steps, as listed in Table 7. Before deriving these steps, the underlying regulatory framework is examined in more detail.

3.1 EASA Concept Paper: Guidance for Level 1 & 2 Machine Learning Applications

The concept paper provides a foundational structure for developing and certifying AI-based systems in aviation, introducing a dedicated assurance approach designed for the specific characteristics of data-driven ML systems and complementing traditional development practices.

3.1.1 Positioning within the EASA Learning Assurance Framework

A structured engineering framework is essential when developing AI-based systems, particularly in safety-critical domains such as aviation. Traditional standards such as ED-12C/DO-178C [15, 16] and ARP-4754A [17] assume deterministic and fully specified system behavior and are therefore not sufficient to assure the safety of data-driven ML systems. To address this, EASA proposes a W-shaped development process, illustrated in Figure 1, which provides a structured approach to ensuring transparency, traceability, and trustworthiness throughout the ML lifecycle [1]. While similar concepts have been explored in other domains, the W-shaped process adapts and extends these approaches to align with aviation-specific regulatory requirements [22, 25, 34]. Furthermore, it incorporates additional concepts tailored to the challenges of data-driven ML systems [20]. The objective of learning assurance is to establish trust in the performance and behavior of ML models [31]. This is achieved by ensuring transparency across all data and learning-related processes, as well as enabling the early detection and correction of errors during model development [23, 26]. In addition, learning assurance aims to systematically analyze and constrain the behavior of ML models, thereby reducing the opacity commonly associated with such systems [1]. As illustrated in Figure 1, the left branch covers design and preparation phases from requirements management to model training, while the right branch addresses verification and implementation.

The steps of data management, learning process management, and model training form the technical core of the development of ML-based systems [31]. Errors in these early phases cannot be compensated for later by system tests, as

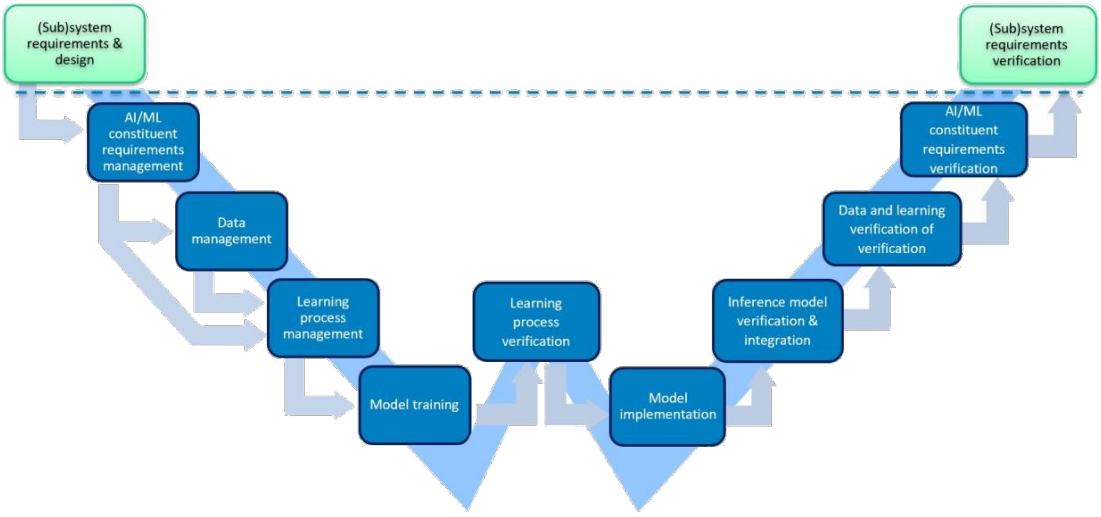


Figure 1: Learning assurance W-shaped process. Steps below the dashed line belong to learning assurance, while steps above correspond to traditional development assurance. Taken from [1].

they directly influence the performance, robustness, and safety of the model [20]. Therefore, EASA emphasizes the importance of high data quality, adequate coverage of relevant operating and environmental conditions, and clearly defined and traceable training processes [1, 26]. Equally crucial is the complete traceability of all data- and learning-related objects [1].

This master's thesis is positioned within the lower-left part of the W-shaped process, focusing on the central learning assurance phases of data management, learning process management, and model training. The AI/ML Constituent Requirements Management step is assumed as a given input, but is not explicitly addressed. Other phases are deliberately excluded, as their inclusion would exceed the scope of this work. The methodology developed in this thesis contributes to structuring these early development phases in a systematic, transparent, and Safety-by-Design-oriented manner, thereby providing a foundation for future ML development processes in aviation.

3.1.2 Conceptual Foundation and Input Elements

The conceptual foundation comprises the input elements that define the context conditions for the ML development process. They originate from the *AI/ML constituent requirements management* phase of the W-shaped process [1, 31] and are treated as given preconditions within the scope of this thesis. The ML model developed in this work is embedded within a larger AI-based aviation system. Therefore, the system-level context must be considered. As illustrated in Figure 2, such systems can be decomposed into traditional and AI-based subsystems [1]. The methodology proposed in this thesis focuses exclusively on the AI-based

subsystem. This distinction is essential, as traditional and AI-based components differ fundamentally in their development logic and assurance requirements [1]. A

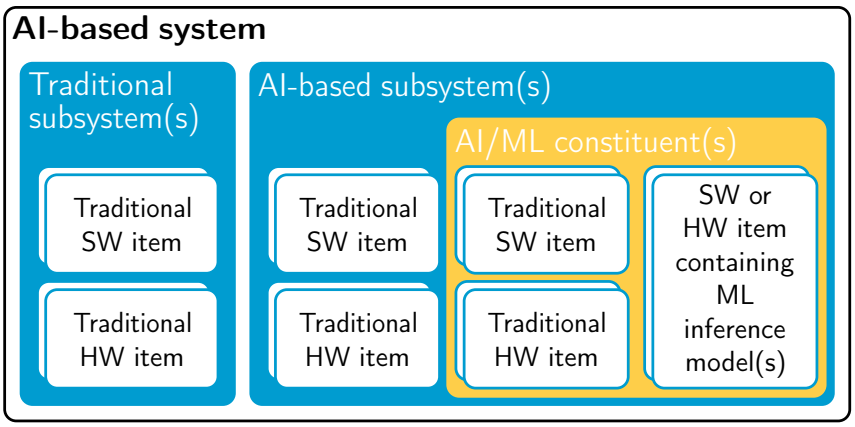


Figure 2: The different components of an AI-based system according to EASA. Taken from [1].

fundamental input element is the Concept of Operations (ConOps) [12, 36]. The ConOps describes the system from an operational perspective, defining its intended use, capabilities, and limitations [1, 63]. It provides the functional context in which the system operates and specifies how the system is expected to interact with its environment and users [36]. In the object detection use case considered in this thesis, ConOps defines the detection of objects on the runway during the landing approach, including the operational scenario, the relevant stakeholders, and the behavior of the intended system under nominal conditions [1, 33].

In addition to the ConOps, the Operational Domain (OD) must be considered [1, 30]. The OD describes the overall operating environment of the complete system, including all external conditions under which the system is intended to function safely [28]. These conditions may include airport characteristics, environmental factors, and technical system constraints [1, 24]. In this thesis, the OD serves as a high-level system context and is not further specified. For an AI-based runway monitoring system, for example, the OD would encompass the entire airport environment, including all runways and taxiways, weather conditions, and interacting systems such as ground vehicles and air traffic control.

The Operational Design Domain (ODD) is derived from the OD and applies specifically to the AI-based subsystem [12, 30, 63]. It defines the concrete operating conditions and parameter ranges under which the AI/ML constituent is expected to function reliably. This includes environmental, geographical, and temporal constraints [1, 31, 33]. Therefore, the ODD establishes the valid deployment conditions of the ML model [34]. In contrast, the ODD for the AI/ML component would narrow this down to the specific conditions relevant for the detection model, such as a defined runway section, daylight operation under clear weather, and a

fixed camera perspective.

Another key input element for the methodology is the definition of Data Quality Requirements (DQRs). DQRs translate qualitative operational constraints into measurable criteria for data collection, preparation, and usage [33]. They ensure that the data used during training, validation, and testing adequately represent the defined ODD and meet the required quality standards [1, 31]. A concrete example is the requirement that all model classes must be represented with a minimum number of annotated instances per split. DQRs are derived from the ODD and form the basis for structured and traceable data management [1, 33].

3.2 Safety-by-Design Methodology for Data Management

The first step of the learning assurance W-shaped process is the *AI/ML constituent requirements management*, which establishes the foundation for the data management methodology. It provides the central input artifacts required for the data management process [1, 31]. The proposed data management methodology is structured into four successive steps, as illustrated in Figure 3. Each of these

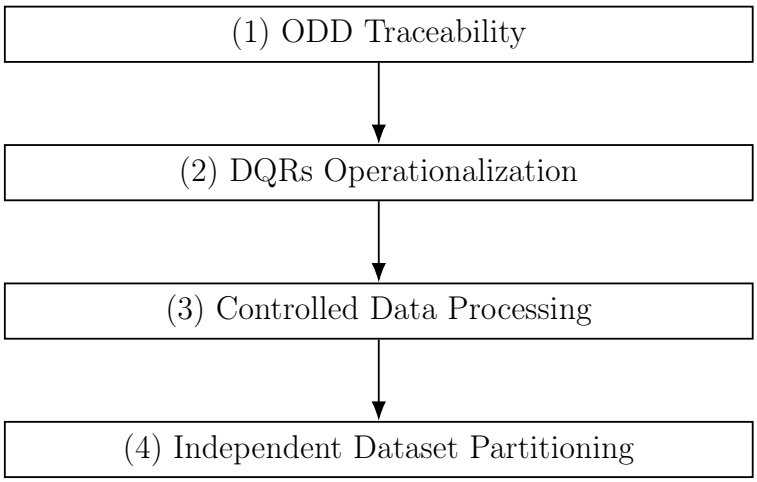


Figure 3: The four steps of the Data Management Methodology.

steps addresses at least one of the previously selected objectives from Table 7 and translates the regulatory requirements into concrete development steps. The order follows the logical dependency between the steps. The ODD must be defined before data can be collected. The collected data must then be verified before it can be partitioned. Finally, the partition must respect the quality criteria established in the DQRs. While the methodology is presented as a sequential process, iterations between the steps may be required to resolve inconsistencies, such as insufficient coverage of the ODD or violations of the DQRs. The individual steps are described in detail in the following subsections.

3.2.1 ODD Traceability

Step (1) establishes a formal, reviewable, and verifiable traceability between the defined ODD and the data used for the learning process [33, 34]. It operationalizes EASA objective DM-01, which requires that relevant data sources are identified and that data are collected in accordance with the defined ODD [1]. The primary output of this step is the ODD-to-Data Traceability Matrix, presented in Table 1, which consolidates all traceability links in a single, structured artifact. The entry shown is an illustrative example.

First, all ODD parameters are extracted from the given ODD description and broken down into discrete units that are each uniquely identifiable. Each unit is assigned a unique ODD identifier (ODD-ID). At this stage, the ODD parameters are not evaluated. They are structured to enable downstream processing [26], as illustrated in Columns (1)–(3) of Table 1. Each ODD parameter is then assessed individually with respect to its relevance for the learning data basis [1, 26]. Specifically, it is determined whether the parameter can influence (i) the ML input data, (ii) the labeling, or (iii) the model behavior [33], resulting in a binary decision (*data-relevant*: yes/no) recorded in Column (4) of Table 1.

For each data-relevant parameter, a data requirement (DR) is derived by specifying which data objects are needed to represent the parameter and which categories or value ranges must be covered to ensure ODD-conform data collection [12, 26, 33]. Each DR is assigned a unique identifier (DR-ID) referencing one or more ODD-IDs, establishing a formal traceability link recorded in Columns (5)–(6). Quantitative quality metrics and acceptance thresholds are not defined at this stage, as these are addressed in Step (2) [1, 26]. Finally, for each DR, at least one data source is identified and documented, including the source type (e.g., real-world, simulation, or synthetic) [36, 39, 41], recorded in Columns (7)–(8). This mapping completes the structured traceability chain:

$$\textit{ODD Parameter} \rightarrow \textit{Derived Data Requirement (DR)} \rightarrow \textit{Data Source}.$$

It provides an initial structural coverage check and enables potential data gaps to be identified, fulfilling the objective of ODD traceability.

3.2.2 DQRs Operationalization

Step (2) operationalizes EASA objective DM-03, which requires that data preparation operations are defined to address the captured requirements, including the DQRs [1]. While Step (1) establishes what data must be collected, Step (2) specifies the criteria by which the collected data shall be evaluated. A fundamental requirement of ODD-compliant data collection is that every relevant ODD

Table 1: Excerpt of the ODD-to-Data Traceability Matrix with an example entry.

(1) ODD-ID	(2) ODD Parameter	(3) Constraint/ Categories	(4) Data-rel.	(5) DR-ID	(6) Derived Data Requirement	(7) SRC-ID	(8) Source Type
ODD-01	Ambient Lighting Condition	Daylight; Night-time operation with standard runway lighting	Yes	DR-01	Image data shall cover daylight and night-time conditions; lighting metadata shall be available.	SRC-01	Internal real-world operational camera data

parameter is represented in the dataset. In other words, the dataset must cover the specified conditions, categories, and value ranges [1, 33].

For each DR from the traceability matrix, potential quality issues are identified by asking: *How would the applicant recognize that this DR is not fulfilled?* This problem-oriented perspective allows the applicant to systematically identify the quality aspects critical for ensuring compliance with the intended operational conditions [33]. For example, DR-05, which requires annotated instances of all defined object classes, could be violated by missing annotations, inconsistent class labels, or an absence of one or more classes in the dataset.

The identified issues are then transformed into concise, unambiguous, and verifiable quality rules [1]. Typically, this results in two to five rules per DR [33]. Although there is no standardized format for DQR formulation, Cappi et al. propose a structured approach in which each DQR is formulated as a testable statement using the pattern: *The dataset shall [measurable condition]* [33]. This approach is adopted in this thesis. For instance, the issue of missing object classes would be translated into the requirement: *The dataset shall contain annotated instances of all defined object classes.*

Each of the DQRs is made measurable by specifying a verification method, a criterion, and a threshold [33]. For the example above, this would be: **Verification method:** count class distribution; **Criterion:** all defined classes present; **Threshold:** at least one annotated instance per class. The objective is not to define complex key performance indicators, but to ensure each requirement can be evaluated in a simple, transparent, and reproducible way [1, 33].

The defined verification rules are applied to the dataset and documented in a structured table including the requirement, verification method, and evaluation result [1]. If one of the DQRs is not fulfilled, the applicant shall not proceed but instead return to Step (1), adapt the dataset accordingly, and repeat the

verification until all DQRs are satisfied [26, 33]. This iterative process ensures that the dataset remains consistent with the defined operational conditions, providing a reliable foundation for subsequent steps in the learning assurance process [1, 31].

3.2.3 Controlled Data Processing

In Step (3), the data shall be prepared for the model training using a predefined, standardized data processing pipeline [26]. The objective of this step is to transform raw data into a consistent, structured, and model-ready dataset [33]. It operationalizes objective DM-03, specifically the requirement to formally define data preparation operations [1], as listed in Table 7. To achieve this, the applicant applies a fixed, predefined set of data processing operations. The systematic and deterministic execution of these steps ensures that the resulting dataset is suitable for further machine learning tasks, while also ensuring that the process remains reproducible and traceable [24, 39].

The preprocessing pipeline consists of the following steps:

1. Data Preprocessing (Cleaning and Preparation)
   (a) removal of faulty or corrupted data (e.g., unreadable images)
   (b) removal of duplicate data
   (c) handling of missing values (e.g., removal or imputation)
   (d) standardization of data formats (e.g., image formats, data types)
   (e) harmonization of units (e.g., meters, seconds)
   (f) cleaning and standardization of labels (e.g., consistent class definitions)
2. Data Transformation
   (a) scaling (e.g., image resizing)
   (b) data normalization (e.g., pixel value normalization)
   (c) feature engineering (not applicable for all ML approaches)

Depending on the application context, additional data processing steps may be required in specific cases. However, the above set of operations provides a sufficient basis for the structured and controlled preparation of data for model training. To ensure a consistent and reliable data processing procedure, all data processing steps shall be executed in a clearly defined order. The steps are performed in the order presented above, ensuring that any data quality issues are resolved before structural and semantic transformations are applied. It is also essential that the applied data processing operations do not invalidate previously defined ODD constraints or DQRs [26, 33]. In particular, preprocessing operations such as resizing, normalization, or augmentation may alter the statistical properties of the data. The applicant shall therefore verify that all ODD parameters remain adequately represented after preprocessing, as transformations can shift the effective data distribution [1]. Where applicable, the developer shall apply the

same processing operations to the ODD specification itself. For example, updating pixel-based thresholds or bounding box dimensions after image resizing. This ensures that the processed dataset remains aligned with the intended operational context and quality specifications and constraints of the ODD in the transformed data space [1, 26]. The input to this step is raw data collected from the identified data sources. The output is a processed, model-ready dataset with a documented and reproducible transformation pipeline.

3.2.4 Independent Dataset Partitioning

The final Step (4) addresses objective DM-06, which requires partitioning the dataset into training, validation, and test subsets. This step is performed once the final dataset has been established, since the independence and integrity of these subsets rely on the previously defined data. Each data sample is uniquely assigned to exactly one subset, ensuring that no data is reused across multiple subsets [24]. This creates a clear separation of the data, forming the basis for a controlled and verifiable learning process.

The partitioning ratio shall be chosen according to established practices, ideally backed by state-of-the-art research or systematic studies. Commonly recommended ratios include 70 / 15 / 15 and 80 / 10 / 10 for training, validation, and test data [49, 64]. This ratio ensures that there is sufficient data for later model optimization while also reserving an adequate sample size for unbiased evaluation [49]. Although there are common split ratios, the optimal split ratio always depends on the use case, the size of the total dataset, and the distribution of classes and representation requirements defined by the ODD. Therefore, developers shall evaluate different split ratios empirically to identify the configuration that best balances model generalization and evaluation reliability [49, 64].

Additionally, the partitioning strategy must ensure the independence of the three subsets, with particular attention to the test dataset [24, 32]. In particular, correlated or dependent data, such as sequences or samples originating from the same scene, must not be distributed across different subsets [62]. The test dataset is strictly isolated from both the training and validation datasets to ensure an unbiased evaluation of the model [24]. Each subset must individually comply with the previously defined Data Quality Requirements. This ensures that all subsets accurately reflect the relevant operational conditions and data characteristics [32]. It is important to balance independence with representation. This involves avoiding biased distributions of scenarios or classes across subsets while ensuring each subset accurately reflects the intended operational context [24, 26, 33].

3.3 Safety-by-Design Methodology for Learning Process Management

In the learning assurance process, data management is followed directly by learning process management. The proposed methodology for learning process management comprises three successive steps, as illustrated in Figure 4. They cover the definition of the model architecture, the specification of the learning process, and the definition of the training environment. Each step is directly linked to one or more of the selected EASA objectives. The steps follow the logical sequence of the corresponding EASA objectives.

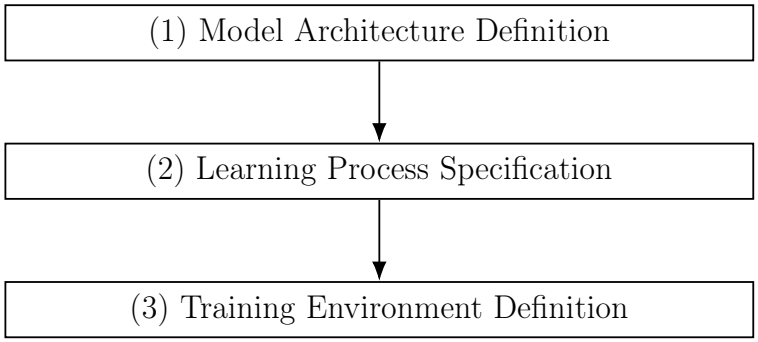


Figure 4: The three steps of the Learning Process Management Methodology.

3.3.1 Model Architecture Definition

Step (1) establishes the model architecture on which the subsequent learning process is based. It operationalizes objective LM-01, which requires a description of the ML model architecture, and contributes to LM-02 through the selection and justification of the model family [1]. First, the applicant derives the requirements the ML model shall fulfill by selecting the task category that best represents the use case [26]. Typical categories include classification, regression, object detection, segmentation, anomaly detection, and sequence modeling [48, 49, 55, 65]. Based on the selected category, the input data types and expected output format are defined, alongside relevant performance and operational requirements such as robustness, computational efficiency, latency, and scalability [1, 26, 66]. Particular attention shall be given to explainability and transparency requirements in safety-critical contexts [1, 27, 32].

Based on the defined requirements, a suitable model family is selected by evaluating alternative approaches. These approaches may include convolutional neural networks, transformer-based models, classical machine learning models, or hybrid approaches [49, 50, 52]. The selections shall consider trade-offs between accuracy, computational complexity, interpretability, and deployment constraints [9, 51, 62].

A specific model architecture is then identified and described in detail, including its structural components such as the input representation, processing layers, and output configuration [50, 51]. Where applicable, the architecture is decomposed into functional modules (e.g., backbone, intermediate layers, output heads) [50, 55, 58]. The output heads then function as the prediction mechanism for object localization and classification [58, 64]. The data flow is specified explicitly to ensure reproducibility [1, 26].

Finally, the chosen architecture is critically evaluated with respect to its suitability for the task, ability to handle the given data characteristics, and performance-related strengths, as well as its limitations regarding sensitivity to data quality, potential failure scenarios, and interpretability constraints [1, 26]. The explainability characteristics of the model are discussed, and the choice of architecture is justified by demonstrating its alignment with the previously defined requirements [1, 12, 27].

3.3.2 Learning Process Specification

Step (2) specifies the learning process of the selected model, fulfilling objective LM-02 [1]. In line with regulatory expectations, the learning process shall be fully specified before implementation to guarantee transparency, traceability, and auditability of all design decisions [1].

The process begins with selecting an appropriate learning paradigm and algorithm. Depending on the problem formulation and the availability of labeled data, different options, such as supervised, unsupervised, or reinforcement learning, may be used. Supervised learning approaches are often favored for perception-based tasks due to their ability to leverage annotated datasets [9, 48].

After selecting the algorithm, the functional components of the learning model shall be defined in detail. This includes the specification of activation functions, which introduce non-linearity to the model and directly affect training stability and convergence behavior [67]. The stability and convergence properties of the chosen activation functions shall be considered, as certain functions may exhibit failure modes such as gradient vanishing or neuron inactivation that can compromise training reliability [67].

A critical component of the learning process specification is the definition of the loss function. This determines the objectives of the optimization process during training and therefore directly influences the behavior of the resulting model. More specialized loss formulations may be required for complex tasks such as object detection. The chosen loss function shall align with the evaluation metrics used to assess model performance. For example, employing Intersection over Union (IoU)-based losses ensures that the training objective directly reflects performance

measures such as mean Average Precision [65, 68]. In tasks with significant class imbalance, techniques such as Focal Loss can be considered to ensure that rare but safety-critical objects receive adequate attention during training [65, 68].

In order to enable a structured evaluation of the learning process, relevant performance metrics shall be defined. In the context of supervised learning, this involves characterizing bias and variance to assess underfitting and overfitting behavior [1, 59]. Additionally, robustness metrics shall be specified to evaluate model performance under different environmental conditions [27, 38].

Another essential element is the definition of hyperparameters, including the learning rate, batch size, number of training epochs, and regularization parameters [9, 61]. Selecting the right hyperparameters has a significant impact on both model performance and training stability [69]. The search space shall be defined based on domain knowledge or empirical default values, as an unstructured search space can exclude optimal configurations or waste computational resources [69]. While grid search becomes infeasible for high-dimensional parameter spaces, random search and Bayesian optimization offer more efficient alternatives—the latter being particularly suitable for expensive model training processes [70, 71]. To avoid optimistic bias in model evaluation, hyperparameter tuning shall be performed exclusively on the validation dataset, while the test dataset remains strictly isolated for final performance assessment [66, 69].

In addition, the initialization strategy for the model parameters shall be specified. The choice between random initialization and using pretrained weights substantially impacts convergence speed and final model performance [59]. Transfer learning is often preferred in practical applications because it leverages prior knowledge and reduces the necessary training data [9]. Pre-trained weights from generic datasets may introduce domain bias when applied to specific environments, making full fine-tuning of the entire network essential to adapt learned representations to the target domain [38, 59]. When fine-tuning a pre-trained model, a phased training approach is recommended: first training only the task-specific output layers while keeping the pre-trained backbone frozen, followed by full network fine-tuning [59, 61]. Regardless of the chosen approach, the initialization strategy shall be explicitly documented and justified. The outcome of this step is a complete and traceable specification of the learning process that forms the basis for the subsequent training phase and ensures compliance with the learning assurance framework.

3.3.3 Training Environment Definition

Step (3) of the learning process management methodology addresses the definition of the training environment in which the learning process is executed. In accordance

with objective LM-02, the training environment shall be clearly identified to ensure transparency of the conditions under which the model is developed [1]. As existing regulatory guidance provides only limited detail on how this objective should be operationalized, this step introduces a structured approach for specifying the training environment, including the hardware and software components required for reproducibility, verification, and certification.

The hardware environment is defined by specifying all computational resources involved in the training process. This includes, in particular, the type and configuration of graphics processing units (GPUs), central processing units (CPUs), system memory (RAM), and storage components [1, 26]. These elements directly influence the feasibility and performance of the training process, as they determine factors such as computational capacity, training duration, and the range of possible training configurations [1]. For instance, the available GPU memory constrains the maximum batch size, while the overall computational performance affects convergence time [9, 61]. A documentation of the hardware setup ensures that the training process is reproducible under equivalent conditions [1, 26].

The software environment shall be fully specified, including the operating system, machine learning frameworks, and all relevant libraries and dependencies required for model training [1, 26]. Particular importance is given to the explicit definition of software versions, as variations in frameworks or libraries may lead to differences in training behavior and model performance [41, 59]. By establishing a consistent and well-defined software stack, unintended variability is minimized, thereby ensuring comparability and reproducibility of results [1].

The training configuration is defined according to the specified training environment. Unlike the hyperparameter values defined in Step (2), the training configuration defined here focuses on the constraints imposed by the environment, like what is computationally feasible given the available hardware. This includes parameters that are directly influenced by the computational setup, such as batch size, number of training epochs, and the degree of parallelization during data loading and processing [41, 59, 61]. They are not defined in isolation but are explicitly linked to the capabilities and constraints of the hardware and software environment [9, 41]. This ensures a stable training process and is reproducible when executed under the same conditions [9, 57]. The result is a structured specification of the training environment, covering hardware, software versions, and configuration parameters required for reproducibility and later verification [1, 26].

3.4 Safety-by-Design Methodology for Model Training

The proposed model training methodology comprises two steps, as illustrated in Figure 5. Both steps operationalize objective LM-05, which requires the documentation of model training results [1]. The first step covers the execution of the training process, while the second addresses its documentation. The order of the steps reflects the logical structure of the training process.

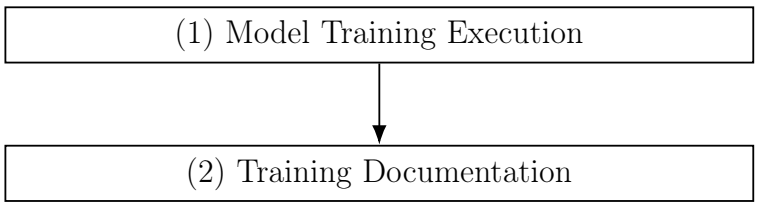


Figure 5: The two steps of the Model Training Methodology.

3.4.1 Training Execution

Step (1) of the model training methodology performs the execution of the training process. While EASA objective LM-02 specifies the expected process, LM-05 requires documentation of the results. The execution itself is the connecting step between these [1]. This step produces the trained model and performance indicators, which serve as input for the documentation step.

During training, the model's performance is continuously monitored. This includes tracking the loss progression as well as relevant performance metrics, such as precision, recall, or mean average precision (mAP), depending on the application. These metrics provide an initial indication of the model's learning behavior and its ability to capture patterns within the training data [1, 59].

It is important to note that this step only operates on the training dataset. The validation and test datasets are not considered at this stage, as they are reserved for subsequent evaluation steps [1, 69]. This separation ensures that the training process remains focused on parameter optimization.

The training process is executed iteratively. Based on the results, adjustments to the learning process management can be made. These may include changes of hyperparameters, model configuration, or data-related aspects. Whenever such modifications are introduced, the training execution shall be repeated to ensure that the model is consistently trained under the updated conditions [9, 59].

The training process shall continue until convergence criteria are met, such as reaching a plateau in validation loss or until the predefined number of epochs is reached. Early stopping may be applied to prevent overfitting [59]. The result is a trained model with learned parameters and initial performance indicators, which sets the basis for the subsequent evaluation in Step (2).

3.4.2 Training Documentation

Step (2) of the model training methodology focuses on the documentation of the training results. In accordance with objective LM-05, the outcomes of the model training shall be recorded to ensure transparency, traceability, and reproducibility of the learning process [1]. This step focuses on the structured documentation of the applied configurations and the resulting model performance. It does not introduce new design decisions, but captures how the previously defined architecture, learning process, and training environment were applied in training.

The documentation includes all relevant information required to reproduce and evaluate the training process. This includes the applied training configuration, such as the selected hyperparameters, optimization strategy, loss function, and number of training epochs, as well as the corresponding model version [41, 59, 61]. Documenting the actual configuration used during training ensures consistency between the defined learning process and its implementation can be verified [1].

In addition to the training configuration, the achieved training results shall be documented [1, 26]. This includes the progression of the loss function throughout training, as well as relevant performance metrics as described in the model training execution [9, 38]. These results provide a transparent representation of the model's learning behavior and set a basis for subsequent evaluation and validation steps [41, 59].

The documentation shall reflect the iterative nature of the training process. When performing multiple training runs, each iteration shall be recorded along with the corresponding changes. This enables a comparison of different training configurations and justifies the final model selection [38, 59]. The result is a structured training documentation that enables reproducibility and subsequent evaluation. This documentation contributes to the overall traceability of the AI-based system.

The presented methodology implements selected objectives of the EASA Concept Paper and provides structured guidance. Since only a subset of the EASA objectives is addressed, the methodology does not constitute a complete assurance process. Instead, it covers data management, learning process management, and model training of the assurance process for AI-based systems. Fulfilling the selected objectives is a necessary prerequisite for the learning assurance process and provides a structured foundation for the safe development of ML-based aviation systems.

4 Implementation and Validation

The following chapter presents the application of the previously developed Safety-by-Design methodology to a concrete use case of runway object detection. The aim is to demonstrate the practical applicability of the methodology and to validate its ability to guide the development of AI-based systems in compliance with EASA requirements. All three phases of data management, learning process management, and model training are covered. Each step is applied, and the outcome is documented to ensure traceability and reproducibility.

4.1 Use Case Definition & System Context

The use case presented in this thesis is the automatic detection of FOD and other safety-critical objects on runways. As described in Section 1.1, such objects on the runway pose a significant safety risk to flight operations [3]. The system's goal is to assist the flight crew during the final approach. If a hazard is detected, the crew receives the information in time to initiate a go-around maneuver if necessary. In accordance with the EASA level classification, this system is therefore categorized as a Level 1 AI application. It provides detection support while the flight crew maintains full authority over any resulting actions [1].

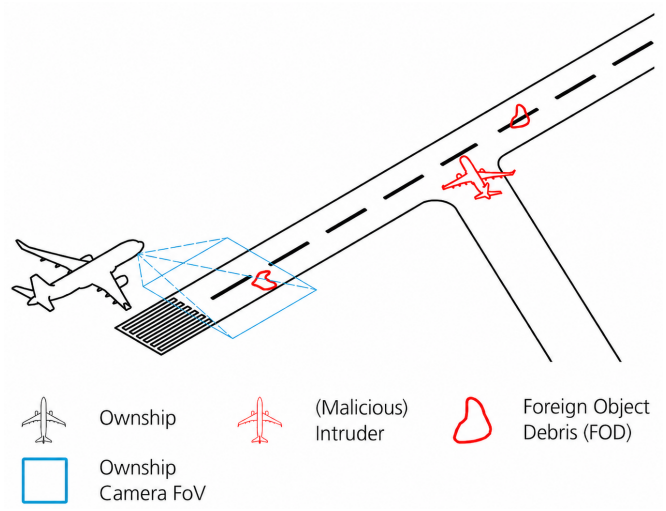


Figure 6: Operational concept of the runway object detection use case, illustrating the aircraft approach scenario with camera-based detection of foreign object debris on the runway.

The described use case is illustrated in Figure 6. An aircraft, referred to as the ownship, is on final approach to the runway. As shown, a camera is mounted on the underside of the approaching aircraft, and its field of view (FoV) covers the runway surface ahead. It continuously captures images of the runway during the final landing approach. The recorded images are supplied as the input elements

to an AI-based object detection system. This component detects and classifies objects located on or near the runway. Detection targets include aircraft, ground vehicles, persons, drones, and various categories of FOD.

In accordance with the system architecture proposed by EASA and introduced in Section 3.1.2, the overall system can be divided into a traditional subsystem and an AI-based subsystem [1]. On the one hand, the traditional subsystem comprises the aircraft itself, the aviation electronics techniques, and the camera hardware. The AI-based subsystem, on the other hand, is the perceptual component that processes camera images and outputs bounding box predictions with corresponding class labels. Since this work focuses exclusively on the AI-based subsystem, the traditional subsystem is assumed to be given and is not the subject of this work.

The following documents the application of the methodology developed in Section 3 to this use case. First, the simulation-based dataset is created. It is a synthetic dataset for runway object detection developed specifically for this work. It is created in accordance with EASA objectives DM-01, DM-03, and DM-06 Section 3.2. Second, the development and training of the YOLO11m object detection model is described in accordance with objectives LM-01, LM-02, and LM-05 Section 3.3 and Section 3.4. Each development step is documented and assigned to the corresponding EASA objective.

4.2 Experimental Setup

The experimental setup consists of two parts. The creation of the dataset in a simulation environment and the training of the object detection model on GPU hardware. Unreal Engine 5.3 (UE) is a photorealistic, real-time rendering engine developed by Epic Games [72]. ProjectAirSim (AirSim) is an open-source plugin built on UE5 that is designed for simulating aircraft and autonomous vehicles [73].

Figure 7 illustrates the simulation-based data generation pipeline. UE provides the visual simulation environment and photorealistic airport model, and AirSim handles the aircraft's approach. Together, they enable fully controlled generation of annotated camera images from the perspective of a landing aircraft. Bounding box labels are created automatically through a 3D-to-2D projection ensuring that every image in the dataset has a precise and consistent annotation [39, 41].

The motivation for using synthetic data in this work is the infeasibility of collecting real-world data for this use case, as discussed in Section 2. Comprehensive annotated datasets of runway objects captured during real landing approaches do not exist [3, 9, 46]. The simulation directly addresses this gap and offers three key advantages. Precise control over environmental conditions such as lighting and weather, ensuring systematic ODD coverage [29, 36, 39]. Full configurability

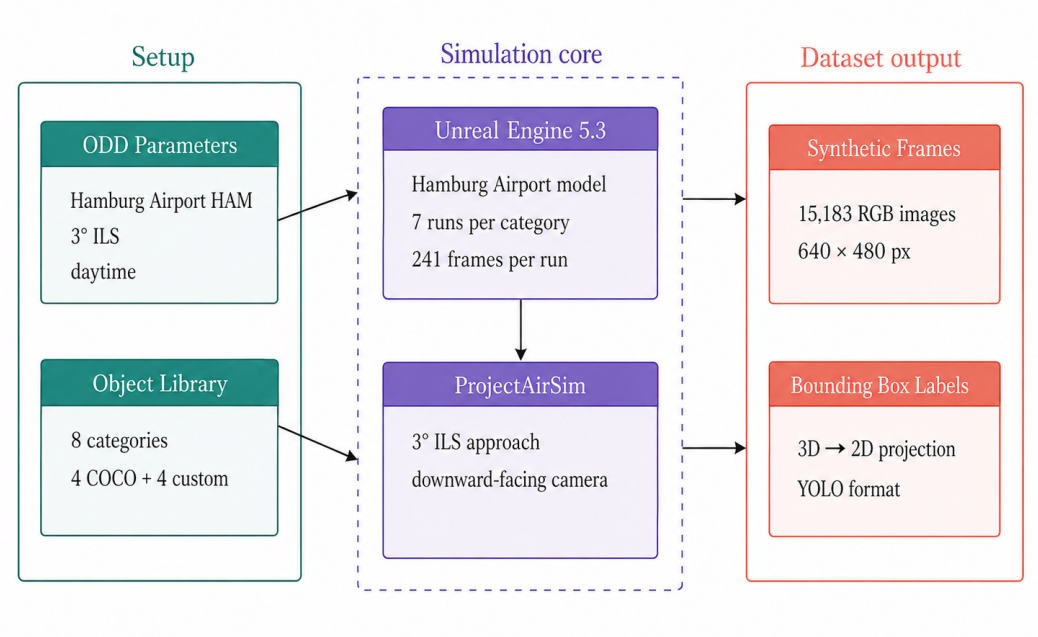


Figure 7: AirSim pipeline for synthetic data generation, illustrating the creation of image frames and YOLO labels using Unreal Engine 5.3 and ProjectAirSim.

of object placement, flight path, and camera settings across multiple runs [40, 59] and fully automatic bounding box annotation through 3D-to-2D projection, eliminating manual labeling effort [39, 74]. However, the use of simulation introduces a simulation-reality gap that must be considered when interpreting the model's transferability to real-world operations [38].

In addition to the simulation environment, the experimental setup includes hardware and software, listed in Table 2. The training is performed on a server at the German Aerospace Center (DLR), providing an NVIDIA GeForce RTX 4090 GPU. The Ultralytics framework (version 8.3.233) is installed for training, providing a standardized implementation of the YOLO model architecture. The YOLO11m model is initialized with pretrained weights from the COCO dataset [60].

Table 2: Hardware and Software environment used for model training.

Component	**Version**	**Role**
NVIDIA RTX 4090	24 GB VRAM	Training GPU
Python	3.10.12	Runtime environment
PyTorch	2.9.1+cu128	Deep learning framework
Ultralytics	8.3.233	YOLO model implementation
CUDA	12.8	GPU computing
cuDNN	9.10.2	GPU neural network library
OpenCV	4.11.0	Image processing
NumPy	2.2.6	Numerical computation

Figure 8 shows the complete data workflow of this study. As shown, the entire experimental procedure is divided into two sections. The section on the left covers data management, including ODD traceability, DQR definition, simulation-based data generation, preprocessing, and dataset splitting. The section on the right comprises two training phases and a final evaluation on the test dataset, which is part of model development. Both sections are described in detail in the following.

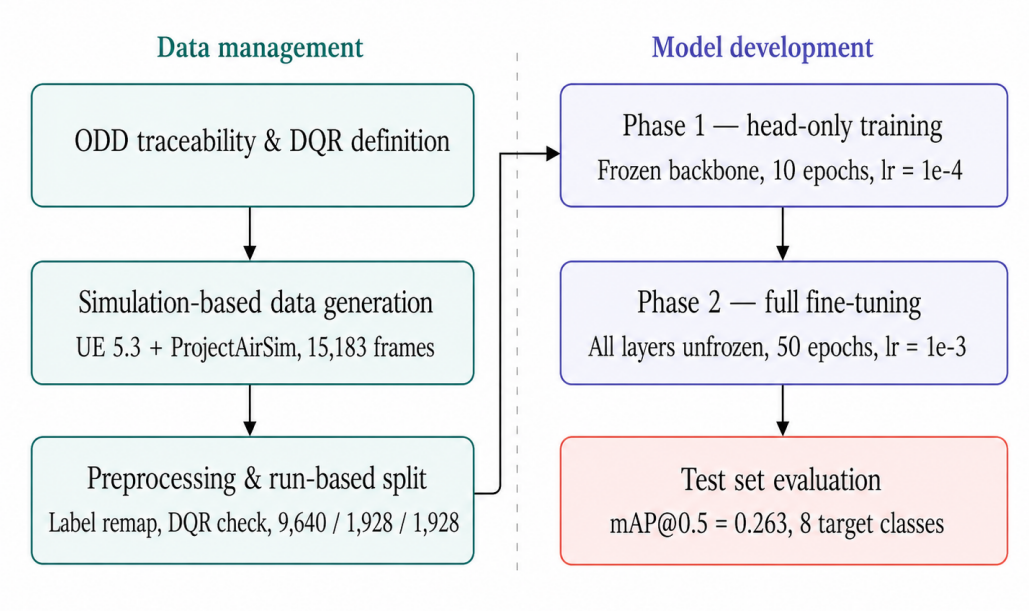


Figure 8: Overall data pipeline of the runway object detection use case, illustrating the process from ODD-based data management and simulation-based dataset generation to model training and evaluation.

4.3 Simulation-Based Dataset Creation

4.3.1 Input Elements

Since the focus of this work is on the AI-based perception component, a system-level ODD is out of scope, the analysis is restricted to the AI/ML constituent ODD defined below. The ODD is a key input for the data generation process that specifies the environmental and scenario-related constraints [1, 33]. In this work, the landing approach on RWY 23 at Hamburg Airport (EDDH) is selected as the operational scenario, as illustrated in Figure 9 where the runway is highlighted. This configuration represents a typical instrument landing scenario and provides a realistic basis for the ODD definition.

The environmental conditions are limited to daytime operations between 10:00 and 14:00 under cloudy weather conditions. This time frame was chosen to ensure stable illumination and to avoid the extreme shadows caused by the low solar angle during sunrise or sunset. Cloudy conditions were chosen because

Figure 9: Unreal Engine implementation of the Hamburg Airport environment shown from an altitude of approximately 2,500 m. The overview illustrates the extent of the simulated airport map and its surrounding urban environment.

they are representative of the selected location and because they reduce harsh shadows, glare, and strong variations in direct sunlight. Such shadows would negatively affect the detection performance. Consequently, the generated images reflect environmental conditions that are both location-specific and suitable for controlled, consistent visibility during dataset generation. The resulting visibility and illumination characteristics are illustrated in Figure 10.

The following entity types are considered within this operational scenario: *airplane*, *container*, *debris_pile*, *debris_scatter*, *drone*, *person*, *suitcase*, and *vehicle*. Multiple 3D asset models represent each class to increase intra-class variety and ensure a more representative dataset [39]. An exception is debris, which is split into two distinct classes, *debris_pile* and *debris_scatter*, due to the visual differences between piled and scattered debris on the runway surface. These entities represent both safety-critical hazards, such as FOD, as well as security-relevant threats, such as unauthorized drones. They provide a comprehensive scope of detection for the landing approach.

The ownship, equipped with a camera system, follows a predefined landing trajectory. The approach starts at a distance of approximately 800 m before the runway threshold at an altitude of approximately 138 m (453 ft) AGL and

Figure 10: Camera perspective during the simulated approach to RWY 23 at Hamburg Airport. The image shows the midpoint of the approach trajectory and illustrates the visual perspective used for image generation.

400 m past the threshold at approximately 24 m (79 ft) AGL. This covers a total trajectory length of 1200 m. The approach speed is assumed to be 66 m s^{-1} (128 kn) on a glide slope of approximately 3.1°. The camera system is mounted on the underside of the ownship at a pitch of −15°, that is, 15° below the horizontal plane. It has a horizontal field of view of 60° and a capture resolution of 1920 px × 1080 px. The resulting camera perspective during the approach is illustrated in Figure 10.

4.3.2 ODD Traceability

In accordance with Step (1) of the Data Management Methodology, as described in Section 3.2.1, this section establishes the traceability between the defined ODD and the data used for the learning process [1]. The goal is to ensure that the generated dataset represents the relevant operational conditions and that all data requirements can be traced back to the ODD parameters. The result of this section is the ODD-to-Data Traceability Matrix, provided in Table 8 in the Appendix.

Based on the above-described ODD, concrete data requirements are derived to ensure that the generated dataset represents the relevant operational conditions. Environmental constraints are translated into data generation parameters. Each is assigned a unique ODD identifier (ODD-ID). The following seven parameters were extracted: the time of day (ODD-01), the weather condition (ODD-02), the airport and runway configuration (ODD-03), the ownship landing trajectory (ODD-04), the ownship speed (ODD-05), the object classes present in the operational environment (ODD-06), and the camera system specification (ODD-07).

Six of the seven parameters were assessed as data-relevant because they directly

affect the visual content of the generated images, the labeling process, or the expected model behavior [1, 26]. The ownship speed (ODD-05) was assessed as not data-relevant, since the model operates on individual image frames and does not perform sequential modeling [1]. The speed, therefore, does not affect the input data, labeling, or model behavior.

For each data-relevant parameter, a concrete data requirement was derived to specify the necessary data objects and value ranges for ODD-conform data generation [26, 33]. The environmental constraints (ODD-01 and ODD-02) result in consistent daytime imagery under cloudy weather conditions. The airport and runway configuration (ODD-03) defines the visual context of the scene. The ownship trajectory (ODD-04) controls the camera perspective throughout the approach, resulting in images that vary in object scale, distance, and viewing angle. To keep it realistic, objects are scaled to reflect their real-world physical dimensions. Variability is introduced through different object instances and positions. The object classes (ODD-06) define the detection targets and directly influence the labeling process. The camera specification (ODD-07) determines the resolution and sampling of all generated images.

Two data sources were identified. The flight simulator (SRC-01) serves as the primary source for all image data. This enables control over all relevant environmental and scenario parameters [37]. The second data source (SRC-02) provides real-world approach data from Hamburg Airport (EDDH) to calibrate the simulated approach path. Using a custom query script, 1329 real-world approaches to RWY-23 were extracted from the OpenSky Network's Trino database [75], covering the period from March 5 to April 5, 2024. For each 500 meter waypoint, median values for altitude, speed, and rate of descent were calculated. The resulting reference trajectory — approximately 305 m (1000 ft) AGL at 65 m s^{-1} (126 kn) — served as the basis for parameterizing the simulated approach path.

4.3.3 DQRs Operationalization

In accordance with Step (2) of the data management methodology Section 3.2.2, the data requirements defined in the ODD traceability matrix are translated into concrete, verifiable quality rules. For each DR, the question, *How would the applicant recognize that this DR is not fulfilled?*, shall be answered. Based on this, the DQRs are formulated, defined, and verified against the dataset. The DQRs, their respective verification methods, and the verification results are listed in Table 9 in the Appendix. The controlled simulation environment satisfies DQR-01 through DQR-04 and DQR-08 directly by design: time, weather, airport scene, trajectory coverage, and image resolution are all fixed parameters in the simulation configuration. DQR-05, DQR-06, and DQR-07 each required corrective action.

The initial verification of DQR-05 revealed that not all eight target classes were represented correctly in the label files. Due to a label ID collision described in DQR-06, some object classes were associated with the wrong class index. Consequently, some target classes appeared absent or mislabeled when the annotation files were inspected. This issue was resolved as part of the label remapping correction described below. After remapping, all eight classes were verified to have annotated instances across all three dataset splits.

During the verification of DQR-06, a label collision was identified between the COCO class index used in the pretrained YOLO11m model and the local class index assigned during the simulation-based generation of the data. Specifically, the COCO dataset assigns 0 as the class *person* and 5 as the class *airplane* [60]. The local simulation assigned 0 to *airplane* and 5 to *person*. Similar collisions happened for the class *drone* and the *vehicle*. Without correction, the model would have learned incorrect class associations during training. The issue was resolved by applying a label remapping script that translated all local simulation indices to the correct target indices across the entire dataset before training. With the correction, DQR-06 was satisfied.

Another problem occurred with DQR-07. The initial verification of DQR-07 revealed a background frame rate of approximately 28 %. This exceeded the defined threshold of 20 %. The elevated rate was due to the additional creation (`empty/` category) data, in which the runway was recorded without any placed objects. These frames were generated to implement negative examples in the dataset. However, their inclusion raised the background rate above the acceptable limit, since the regular data creation already naturally created negative examples. To solve this problem, the `empty/` category was subsequently excluded from the training, validation, and test splits. After this adjustment, the background frame rate was reduced to approximately 18 %, satisfying DQR-07.

After taking corrective action, all eight DQRs were verified as fulfilled. Therefore, the dataset is considered compliant with the defined data quality requirements and is ready for preprocessing and partitioning steps.

4.3.4 Controlled Data Processing

The following covers Step (3) of the data management methodology in Section 3.2.3. It transforms the raw simulation data into a structured and model-ready dataset. The processing steps follow the order defined in Section 3, starting with preprocessing, which cleans and prepares the data, followed by transformation into the target format.

Because the data comes from a controlled simulation environment, preprocessing steps such as removing corrupted or defective images, handling missing values,

and harmonizing different image formats are unnecessary. All generated images are in a consistent format and already fully annotated. The most important preprocessing step is label remapping. This was previously described in the context of checking the DQRs, see Section 4.3.3. In this process, the new local class index from the simulation was mapped to the correct target index of the YOLO model. Additionally, the `empty/` category is excluded from the dataset, as its inclusion raised the background frame rate above the defined threshold of 20 %, as described in Section 4.3.3. Furthermore, the bounding box labels are automatically generated [39, 41]. This happens during the simulation via a 3D-to-2D projection. The three-dimensional coordinates of each object are transformed into image coordinates and saved in the corresponding YOLO format [39, 41]. YOLO format describes each bounding box using five values: class index, normalized center position, and the box's width and height relative to the image [58]. Bounding box annotations are used rather than segmentation masks, as the rectangular format is sufficient for object detection tasks and reduces annotation complexity [48].

During data transformation, no manual image resizing is required. The Ultralytics framework automatically scales images from 1920 px × 1080 px to 640 px × 640 px and normalizes pixel values during training [58] The dataset consists of 56 simulation runs, with each run comprising 241 frames captured at five-meter intervals along a 1200-meter approach trajectory. Each simulation run contains one placed object per frame. This results in a total of 15,183 consistent, labeled frames in a uniform format and a defined background rate of 18 %. The generated dataset forms the foundation for the final step of the data management methodology.

Representative frames from the generated dataset are shown in Figure 11, illustrating one sample per class with the corresponding ground truth bounding box. For the classes *drone*, *person*, *suitcase*, and *container*, the target object covers only two to three pixels due to the long approach distance and their small physical size. This illustrates a fundamental detection challenge for these classes.

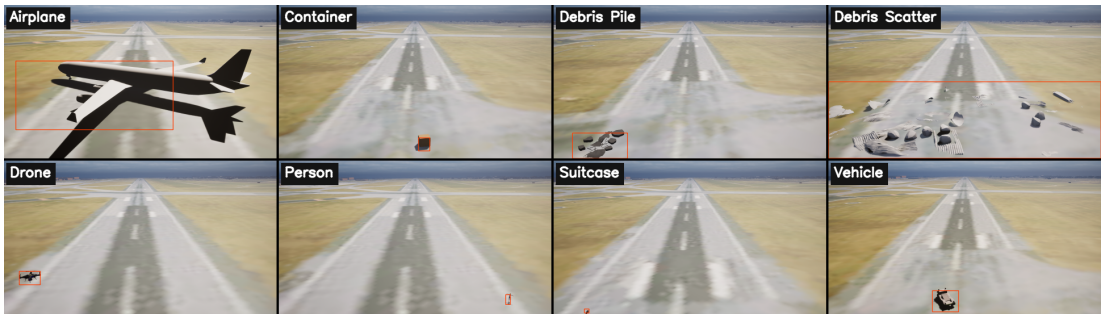


Figure 11: Sample frames from the generated dataset illustrating the result of the simulation-based data generation and automatic annotation pipeline. Each image shows one representative frame per object class with the bounding box.

4.3.5 Independent Dataset Partitioning

The last step in the data management is the division of the dataset into three independent subsets: training, validation, and test data. This step fulfills EASA Objective DM-06 [1], as described in Section 3.2.4. A key dataset design decision is choosing the partitioning unit. Partitioning is performed at the level of simulation runs rather than at the level of individual frames. All 241 frames within a run come from the same approach, involving the same object in the same position on the runway, showing the same object from slightly different distances and angles. Therefore, they are spatially and temporally correlated [62]. A frame-based division would distribute these correlated frames across different subsets [69]. This would lead to information leakage between the training and test data [33, 64]. The run-based division ensures that all frames of an approach are fully assigned to a single subset.

Seven simulation runs represent each of the eight object classes. These runs are distributed across the three subsets at a ratio of 5 / 1 / 1. This reflects a split of 71 / 14 / 14 for training, validation, and testing [64]. Table 3 gives an overview of the splits. The background rate remains consistent across all three splits, covering a range from 15.4 % to 19.5 %, because the background frames are generated by the flight path and distributed evenly across all runs [32].

Table 3: Dataset split overview after run-based partitioning.

Split	Frames	Runs per class	Total runs	Background rate
Train	9640	5	40	18.1 %
Val	1928	1	8	15.4 %
Test	1928	1	8	19.5 %
Total	**13 496**	**7**	**56**	**17.9 %**

The test dataset is strictly isolated from the training and validation data. It is never used during model development or hyperparameter selection. It is reserved exclusively for the final, unbiased evaluation of the trained model [69]. This completes the data management. It meets all the defined DQRs and is ready for the learning process management.

4.4 Model Development

4.4.1 Model Architecture Definition

The model development begins with the definition of the model architecture as described in Section 3.3.1. In order to decide on the model architecture, the use case for the model must first be considered. In this work, the model shall detect

and classify safety-critical objects in camera images of the runway. The use case falls under the category of object detection. For each object detected, the model must output a bounding box and a class label [50]. A simple image classification would be ineffective because the exact location of the object in the image is relevant [48]. Instance segmentation is not appropriate for this use case because pixel-precise contours do not provide additional information for the go-around decision. The model must identify eight target classes and handle a variety of object sizes.

Based on these requirements, the YOLO model family is selected [54]. As discussed in Section Section 2.2.2, YOLO outperforms two-stage and transformer-based alternatives in inference speed and data efficiency [48, 57], making it the established standard for runway object detection tasks [53]. Additionally, the YOLO model family is used for object detection applications at airports and runways and is considered an established standard for this task [9, 55, 56]. Within the YOLO family, there are different variants and versions. Although newer YOLO versions were available at the time of implementation, YOLO11m was selected because it represents the most recent version with sufficient peer-reviewed literature and best practices for fine-tuning and transfer learning [51, 58, 61]. Adopting a version without an established methodology would have undermined the literature-based approach central to this thesis. Among the available variants from nano to extra large, the medium variant YOLO11m was selected for its balance between model size, inference speed, and detection accuracy [58].

Figure 12 shows the architecture of the YOLO11m model in a simplified representation. The model consists of three functional components [58]. On the left side, the backbone, which is based on CSPDarknet, extracts hierarchical visual features from the input image. In the middle, the neck combines these features with a feature pyramid network (FPN) and a path aggregation network (PAN) [50, 58]. This creates a multi-scale feature representation. The detection head processes this representation and generates predictions consisting of class probabilities and bounding box coordinates [58].

The selected YOLO model is already initialized with pre-trained weights from the COCO dataset [51]. The COCO dataset contains 80 categories of common objects [55]. Since COCO and the generated dataset use a different number and type of classes, the detection head is replaced with a newly initialized head with eight outputs, according to the number of classes of the generated dataset. The backbone and neck use the pre-trained COCO weights as a starting point for the fine-tuning. This transfer learning strategy enables stable results on a relatively small dataset, because the backbone's general visual features do not need to be learned from scratch [9, 61]. The model only accepts input images with a

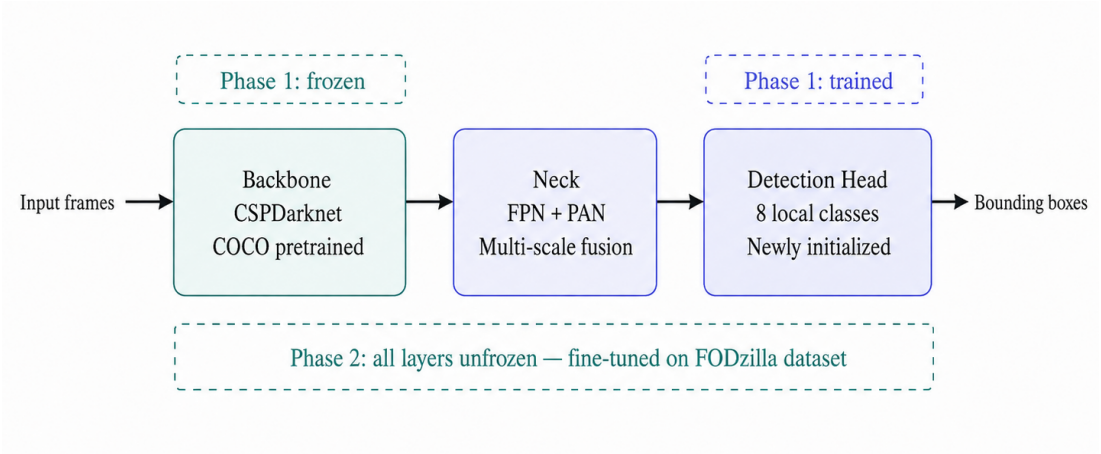


Figure 12: Architecture of the adapted YOLO object detection model, illustrating the pretrained backbone, multi-scale feature fusion neck, and newly initialized detection head for runway object detection.

resolution of 640 px × 640 px [62]. As described in Section 3.2.3, the Ultralytics framework automatically scales and normalizes the original simulation images to the required input format. Overall, YOLO11m is well-suited for this task. Its multi-scale feature architecture covers the full range of object sizes present in the dataset, while the transfer learning strategy reduces the required training data [58]. Notably, like all neural networks, YOLO11m offers only limited explainability, due to its complexity [51, 66]. Individual decisions cannot be traced back directly, which is a relevant in the context of EASA transparency requirements and must be addressed in future verification steps beyond the scope of this work.

4.4.2 Learning Process Specification

As described in Section 3.3.2, this step involves fully specifying the model's learning process before starting with the training. All design decisions are documented in accordance with EASA objective LM-02 [1]. The YOLO model is trained using supervised learning. Each image in the dataset is annotated with a bounding box and class labels [48]. These serve as targets for the optimization process. Supervised learning is a suitable approach for vision-based tasks, such as object detection [6]. Stochastic gradient descent (SGD) is used as an optimization algorithm with momentum combined with a cosine-shaped learning rate decay schedule (cosine annealing) [76]. Additionally, YOLO11m uses SiLU (Sigmoid Linear Unit) as the activation function in the backbone and neck of the model [58]. SiLU was chosen over standard ReLU because it avoids the dying ReLU phenomenon, in which neurons become permanently inactive when their inputs are consistently negative, resulting in zero gradients and no further learning [67].

A key design decision is dividing the training into two phases, as shown in

Figure 12. In Phase 1, only the newly initialized detection head is trained while keeping the backbone frozen. The aim is to stabilize the detection head before adjusting any of the pre-trained backbone weights. Without this stabilization, training all layers in Phase 2 at once would lead to gradient shock, which would destabilize the pre-trained features and disrupt the training process [9, 61]. In Phase 2, all layers are unlocked and trained together. Now, the model can adapt the backbone's pre-trained COCO features to the runway environment's specific visual characteristics. Pre-trained weights from COCO may introduce domain bias when applied to the runway simulation environment. Full fine-tuning in Phase 2 is therefore essential to adapt the learned representations to the target domain [38, 59]. Mosaic and copy-and-paste augmentation are enabled in this phase [51]. They address class imbalances in the dataset and improve the model's ability to handle different object positions and scales [55].

Another important element is the loss function. YOLO11m optimizes three loss components during training [62]. The box loss, $\mathcal{L}_{\text{box}}$, measures the localization accuracy of the predicted bounding boxes using an intersection over union (IoU)-based metric [58]. The classification loss (CLS), $\mathcal{L}_{\text{cls}}$, uses binary cross-entropy to evaluate class predictions. The Distribution Focal Loss (DFL), $\mathcal{L}_{\text{dfl}}$, improves the precision of box boundary estimation through a distribution-based approach. The total training loss, $\mathcal{L}_{\text{total}}$, is the weighted sum of these three components [58].

$$\mathcal{L}_{\text{total}} = 7.5 \cdot \mathcal{L}_{\text{box}} + 0.5 \cdot \mathcal{L}_{\text{cls}} + 1.5 \cdot \mathcal{L}_{\text{dfl}}. \tag{1}$$

For bounding box regression, the Ultralytics implementation uses a CIoU-based loss rather than standard MSE, as it accounts for box overlap, aspect ratio, and center distance simultaneously, providing more stable convergence than squared error metrics [65, 68].

The primary evaluation metric used is mean average precision (mAP) with an IoU threshold of 0.5. If the predicted bounding box overlaps the ground truth object by at least 50 % IoU, a detection is correct [48, 65]. Precision and recall are combined in the precision-recall curve [48]. Average precision (AP) is calculated per class as the area under the precision-recall curve [65]. mAP@0.5 is the average of these values across all eight target classes [48, 51]. mAP@0.5 was chosen as the primary metric for two reasons. First, it is the standard metric in object detection research, allowing for direct comparison with related FOD detection work [53, 56]. Second, mAP@0.5 captures both the classification quality and the localization accuracy of the model in a single value [77]. This makes this metric well-suited for evaluating the overall detection capability of all eight target classes. The threshold of 0.5 was chosen over stricter thresholds such as

mAP@0.75 or mAP@0.5:0.95, as the primary objective is reliable object detection rather than pixel-precise localization [65]. A bounding box that correctly identifies the presence and approximate location of a hazard is sufficient for the go-around decision [53]. Considering the use case, the focus is on safety-critical runway detection. Therefore, recall is particularly important because a missed detection presents a higher risk than causing a false alarm [27]. The mAP@0.5 metric considers recall through the precision-recall curve, reflecting this critical safety aspect of model performance [1].

Table 4 lists the hyperparameters for both training phases. In accordance with the methodology, hyperparameter decisions were evaluated exclusively on the validation dataset. The test dataset was not consulted during any training or tuning decision, ensuring an unbiased final evaluation [69].

Table 4: Hyperparameter configuration for Phase 1 and Phase 2 training.

Hyperparameter	**Phase 1**	**Phase 2**
Epochs	10	50
Learning rate	10^{-4}	10^{-3}
Batch size	32	32
Image size	640 px	640 px
Frozen layers	Backbone (freeze = 10)	None
Mosaic augmentation	0.0	1.0
Copy-paste augmentation	0.0	0.3
Close mosaic (final epochs)	*N/A*	10
Optimizer	SGD	SGD
Random seed	42	42

Table 4 shows the lower learning rate in Phase 1 (10^{-4}), ensuring the detection head gradually converges without compromising the training process's stability. In Phase 2, the learning rate is set to 10^{-3}, which is ten times higher than the one used in Phase 1. The higher learning rate allows layers to adapt more effectively to the runway detection problem [62]. The cosine annealing schedule gradually reduces the learning rate toward the end of each phase, allowing the model to make increasingly fine-grained weight adjustments and settle into a stable optimum [58]. Given the scope of this thesis, hyperparameters were set based on established defaults and literature values rather than systematic optimization [58, 59]. While the methodology recommends Bayesian optimization for hyperparameter tuning [70], this was not applied here as the primary goal is to validate the Safety-by-Design methodology rather than to maximize model performance. Mosaic augmentation combines four images into one frame, while copy-paste places objects from other images into the current scene [76]. This method is helpful for underrepresented classes, such as *drone*, *person*, and *suitcase* [61].

4.4.3 Training Environment Definition

In accordance with EASA Objective LM-02, the training environment is fully specified to ensure transparency and reproducibility of the training process. The hardware and software used are already documented in Section 4.2 and listed in Table 2. Training is performed on an NVIDIA GeForce RTX 4090 GPU with 24 GB of VRAM. The available GPU memory capacity limits the maximum batch size. The batch size of 32 was chosen. This ensures a good balance between memory usage and training efficiency [69]. Larger batch sizes were not possible with the selected image size of 640 px × 640 px without exceeding the available VRAM. The Ultralytics framework (v8.3.233) provides a complete training and evaluation pipeline [51]. It automatically handles image preprocessing, data loading, and calculating loss functions and metrics [69]. A fixed random seed of 42 ensures the reproducibility of all stochastic processes during training [32]. The seed is set identically in both training phases. The full description of the training environment ensures that the training process can be replicated under the same conditions, which is a prerequisite for learning assurance according to EASA [1]. All training environment parameters are documented in a structured, human- and machine-readable format, enabling the generation of evidence required for verification and certification activities [1, 26].

4.4.4 Training Execution & Documentation

This section implements Steps (1) and (2) of the model training methodology, as described in Section 3.4.1 and Section 3.4.2. The training follows the specification in Section 4.4.2. Phase 1 and Phase 2 are carried out sequentially. The best checkpoint from Phase 1 is used as the starting point for Phase 2.

During training, only the training dataset is used for parameter optimization. The validation dataset is used to evaluate the progress at the end of each epoch, but does not influence the weights. The test dataset remains completely isolated in both phases and is only used for the final evaluation in Section 5. Training progress in both phases is continuously monitored and documented. This includes the three loss components (Box, CLS, and DFL) as well as the validation mAP@0.5 for each epoch. The resulting training curves for Phase 1 and Phase 2 are shown in Figure 13 and Figure 14 respectively

In accordance with EASA objective LM-05, all relevant training elements are documented and stored in files [1]. This includes the hyperparameters used in both phases, saved model checkpoints, and training and validation metrics for each epoch. This documentation ensures traceability and reproducibility of the training process, forming the basis for possible model verification.

5 Results

5.1 Descriptive Results

This chapter presents the results of implementing and validating the Safety-by-Design methodology. For this reason, two training experiments were conducted and evaluated on the same restricted test dataset. They are referred to as Phase 1 and Phase 2. As a reference point, the pretrained YOLO11m model is first evaluated in a zero-shot setting. This means that it is evaluated without any additional training on the FODzilla dataset. This evaluation establishes the extent of the domain gap. Here, the model achieves an mAP@0.5 value of 0.0008. This result indicates that a pretrained YOLO model cannot detect runway objects from a landing approach perspective without domain-specific fine-tuning. Two factors contribute to this: first, the significant visual domain gap between generic COCO images and the simulated runway environment, and second, the fact that several target classes are not represented in the COCO dataset and are therefore entirely unknown to the pretrained model [60].

5.1.1 Phase 1

The training follows a two-Phase approach. In Phase 1, only the newly initialized eight-class detection head is trained. The backbone remains frozen during this process. The training runs for 10 epochs with a learning rate of 10^{-4} and a batch size of 32.

YOLO11m optimizes three loss components simultaneously. These are box loss for bounding box localization, classification loss for class prediction, and distribution focal loss for precise boundary estimation. As shown in Figure 13, all three training loss values drop constantly throughout all 10 epochs. Meanwhile, the validation mAP@0.5 increases from 0.07 in the first epoch to 0.18 in the last one. Training proceeds smoothly with no signs of overfitting, as the validation loss decreases consistently alongside the training loss throughout all 10 epochs without divergence [59]. The frozen backbone ensures a controlled learning process, as the model's pre-trained COCO features remain unchanged.

5.1.2 Phase 2

In Phase 2, all network layers are unfrozen and trained together in a full fine-tuning process. Here, the best checkpoint from Phase 1 is used as the starting point. The training consists of 50 epochs with a learning rate of 10^{-3}. Additionally, mosaic and copy-paste augmentation are applied to address the class imbalance in the training dataset, as described in Section 4.4.2.

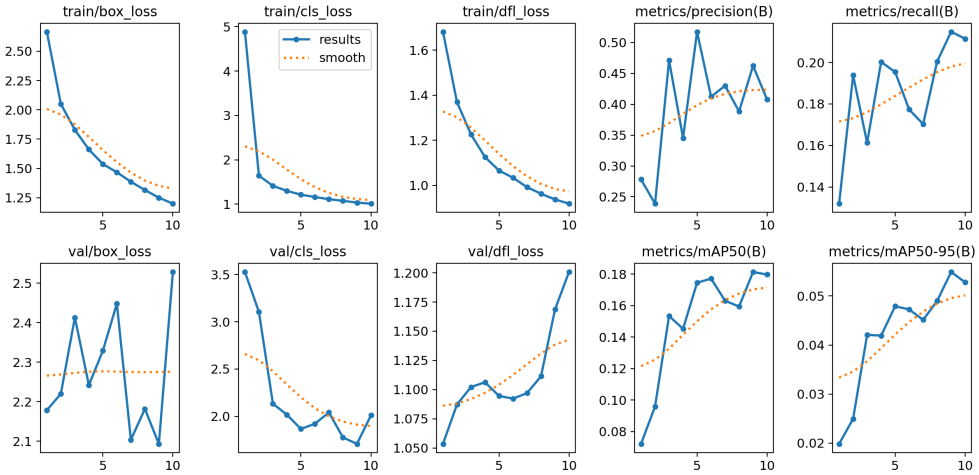


Figure 13: Training and validation metrics for Phase 1 (head-only, 10 epochs).

As shown in Figure 14, it is important to mention that there is a sharp increase in validation classification loss, which rises to nearly 500 in epoch 1 before dropping to almost zero in the following epochs. This spike is caused by the unfreezing of the backbone, which temporarily disrupts the stabilized detection head in Phase 1 due to the sudden release of the backbone weights. This effect disappears quickly and has no lasting impact on the training process, demonstrating that the head stabilization in Phase 1 was successful. If Phase 1 had been skipped, this gradient shock would have negatively impacted the entire training process. Across all 50 epochs, the training losses decrease steadily, and mAP@0.5 rises continuously to 0.25 without reaching a plateau, suggesting that further training could yield additional performance gains. The number of epochs was chosen based on the methodology validation scope rather than model optimization, as discussed in Section 4.4.2.

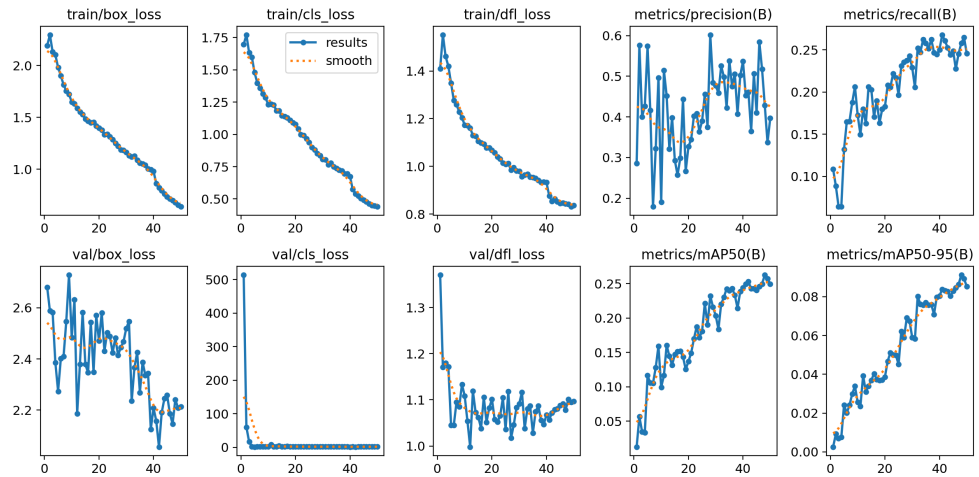


Figure 14: Training and validation metrics for Phase 2 (full fine-tuning, 50 epochs).

5.1.3 Confusion Matrices

The confusion matrix for Phase 1 shows the model's classification behavior after head-only training. In Figure 15, a dominant pattern appears, also known as background absorption. Many of the objects present in the images are not detected and are instead classified as part of the background. This is clear in the case of *drone* (182 out of 188 instances missed), *person* (240 out of 241), and *suitcase* (138 out of 138). For these three classes, object size is an also a contributing factor. The classes with the most correct detections are *debris_scatter* with 110 and *airplane* with 139. However, *airplane* also generates many false positives. In 109 cases, the model incorrectly predicts an airplane in frames that contain only background. This may reflect the model's tendency to associate the runway centerline markings with the elongated shape of aircraft during the early training stage.

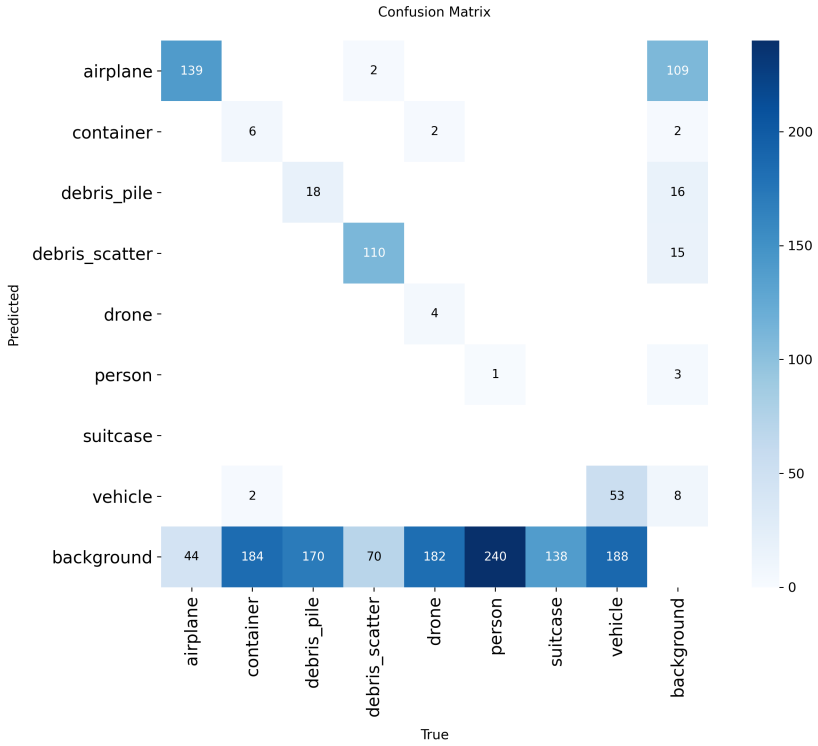


Figure 15: Confusion matrix for Phase 1 evaluated on the test set.

After completing fine-tuning in Phase 2, the results improved significantly for some classes. As shown in Figure 16, the number of correct detections for the class *debris_scatter* increases from 110 to 172. Meanwhile, the number of missed instances drops from 70 to eight. Similar improvements are also seen for *airplane* and *vehicle*. However, the background absorption pattern remains as a dominant error. The classes *drone*, *person*, and *suitcase* are still rarely detected. Notably, in Phase 2, *debris_pile* is frequently misclassified as a *vehicle* (39 cases). This suggests visual similarity between the two classes on the runway

surface, potentially due to comparable shapes and color profiles in the simulation environment. Containers are virtually never detected in either Phase.

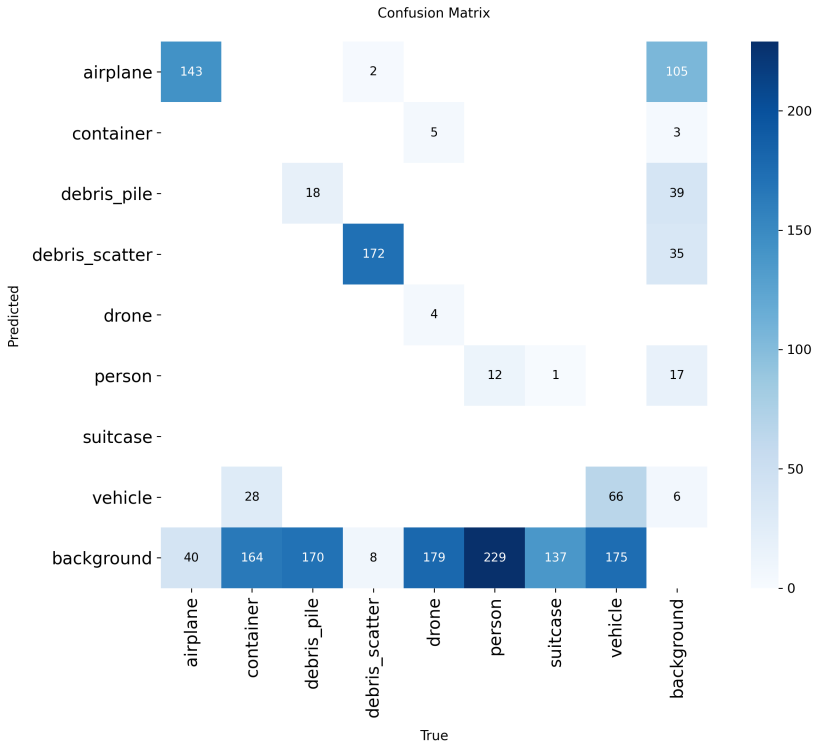


Figure 16: Confusion matrix for Phase 2 evaluated on the test set.

5.2 Estimation Results

Table 5 compares the overall metrics for both Phases on the test dataset. Phase 2 improves the mAP@0.5 from 0.205 to 0.263, representing a relative improvement of 27.8 %. The stricter mAP@0.5:0.95 metric improves from 0.066 to 0.082, confirming the trend across tighter IoU thresholds. Recall improves from 0.205 to 0.252, while precision drops slightly from 0.574 to 0.561. This change is due to the fact that the Phase 2 model produces more detections and thus finds more objects. However, it also produces slightly more false positives.

Table 5: Overall performance comparison between Phase 1 (head-only training) and Phase 2 (full fine-tuning) on the test dataset.

Metric	**Phase 1 (Head-Only)**	**Phase 2 (Full Fine-Tuning)**	**Δ**
mAP@0.5	0.205	0.263	+0.057
mAP@0.5:0.95	0.066	0.082	+0.016
Precision	0.574	0.561	−0.013
Recall	0.205	0.252	+0.047

Table 6 presents the class-level results for all eight target classes. The results reveal two clearly distinct performance groups. The first group includes *debris_scatter*, *airplane*, and *vehicle*. These three classes benefit most from full fine-tuning in Phase 2, achieving mAP@0.5 values of 0.954, 0.547, and 0.417. Their visual appearance is sufficiently distinct, and they also appear frequently in the training dataset to learn stable features. Secondly, there is a group consisting of a *container*, *debris_pile*, *drone*, *person*, and *suitcase*. Their Phase 2 mAP@0.5 values range from 0.012 (*container*) to 0.059 (*drone*). For *container* and *debris_pile*, the value even drops slightly compared to Phase 1. Thus, full fine-tuning mostly improves classes that were already identified in Phase 1. Classes with few training examples or low visual distinctiveness benefit very little.

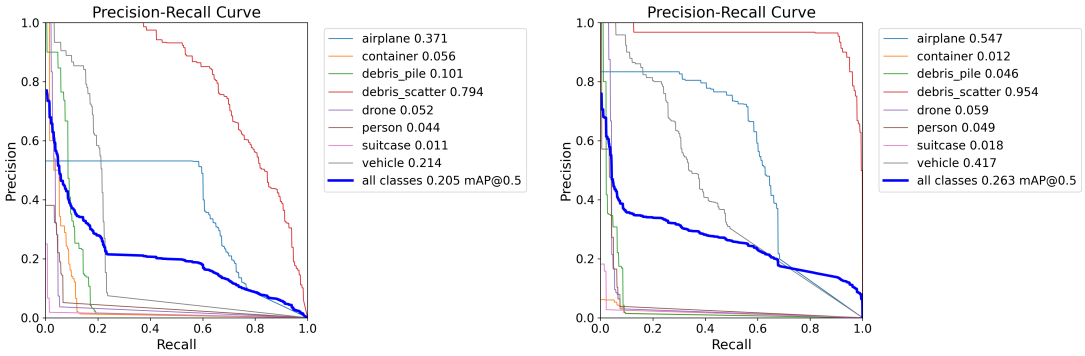
Table 6: Per-class mAP@0.5 comparison between Phase 1 and Phase 2 on the test dataset.

Class	**Phase 1 mAP@0.5**	**Phase 2 mAP@0.5**	**Δ**
airplane	0.371	0.547	+0.176
container	0.056	0.012	−0.044
debris_pile	0.101	0.046	−0.055
debris_scatter	0.794	0.954	+0.160
drone	0.052	0.059	+0.007
person	0.044	0.049	+0.005
suitcase	0.011	0.018	+0.007
vehicle	0.214	0.417	+0.203

A contributing factor to the low detection performance of classes such as *suitcase* and *drone* is the small physical size of these objects. At the typical approach distance of several hundred meters, these objects occupy only two to three pixels at most in recorded images. In most cases, they cannot even be detected by a human observer. This is a fundamental optical limitation rather than a model one. To address this issue, the image resolution would need to be higher, or a multi-sensor setup involving several cameras placed directly at the runway would need to be used. This setup would be capable of recording small objects from a closer distance.

The precision-recall curves for Phase 1 are illustrated in Figure 17a and those for Phase 2 in Figure 17b. In both Phase 1 and Phase 2, the areas under the curve are largest for *debris_scatter*, *airplane*, and *vehicle*. All other classes drop to nearly zero immediately after reaching a low recall value. A comparison of the two curves clearly shows the improvement from Phase 1 to Phase 2, especially

for *debris_scatter* and *vehicle*. Both their curves shift significantly toward the upper-right quadrant.



(a) Precision-Recall curves for Phase 1 evaluated on the test set (mAP@0.5 = 0.205).

(b) Precision-Recall curves for Phase 2 evaluated on the test set (mAP@0.5 = 0.263).

Figure 17: Precision-Recall curves for both Phase 1 and Phase 2.

In the context of safety-critical runway monitoring, recall is of particular importance, as a missed detection poses a greater risk than a false alarm. The classes *drone* and *suitcase* deserve particular attention In Phase 2, both classes have a precision of 1.0 and a recall of almost zero. This means that when the model reports a detection, it is reliably correct. However, there are almost no detections. This is problematic for FOD detection on a runway. In this use case, an undetected object is more dangerous than a false alarm, which would only result in a go-around. Therefore, high recall is more critical to safety than high precision. The model's conservative behavior is due to the remaining 18 percent of background frames in the dataset. Background frames occur when the placed object falls outside the camera's field of view at a given waypoint along the approach trajectory, or when the object is too small to be visible at large approach distances. As a result, the model learns to avoid detections when uncertain, which increases precision but reduces recall.

Overall, the presented results show that applying the full Safety-by-Design methodology in Phase 2 leads to a measurable improvement in detection performance. The structured data management, learning process management, and model training in accordance with the corresponding EASA objectives provide a transparent and reproducible foundation. At the same time, the results also highlight the limitations of the current approach, particularly for rare and visually ambiguous classes and the high number of background images. These findings will be discussed in more detail in Section 6.

6 Discussion

The following chapter discusses the results of this thesis in the context of the research question, identifies limitations, and outlines the theoretical and practical contributions of this work.

6.1 Summary

This thesis addresses the following research question: *What should a Safety-by-Design methodology, in line with selected objectives of the* EASA Guidance for Level 1 & 2 Machine Learning Applications*, look like to develop a simulation-based dataset and a prototype AI model for the detection and classification of safety-critical runway objects (FOD)?* The thesis makes two contributions: the development of the methodology itself and its validation through a specific use case.

Regarding the methodology, six EASA objectives were selected and translated into a structured process of concrete development steps. The methodology is positioned in the lower left quadrant of the W-shaped learning assurance process, covering the phases of data management, learning process management, and model training of a structured development process [1]. It closes the gap between regulatory requirements and their practical implementation.

In a second step, the developed methodology was implemented and validated, by applying to an FOD detection on a runway use case. This application is divided into two parts. In terms of data management, a synthetic dataset was generated, consisting of 13 496 annotated frames from 56 simulation runs across eight safety-critical object classes. The dataset was created using Unreal Engine 5.3 and ProjectAirSim. During the DQRs verification, two quality issues were identified and resolved. These included a label ID collision and an elevated background frame rate. This directly demonstrates the effectiveness of the iterative quality assurance process from the methodology. In the model development process, a YOLO11m model was trained on the synthetic dataset following a two-stage training approach. Starting with a zero-shot mAP@0.5 of 0.0008 with practically no detection capability, the model reached 0.205 after Phase 1 (head-only training) and 0.263 after Phase 2 (full fine-tuning), a 27.8 % relative improvement over Phase 1. Compared to the zero-shot baseline of 0.0008, the final model represents an improvement of more than two orders of magnitude. Visually distinct classes achieved significantly higher mAP@0.5 values than small or underrepresented classes. This suggests that detection performance fundamentally depends on object size and training representation. Some classes regressed in Phase 2, indicating that overall improvement does not reflect consistent improvement across all classes.

6.2 Limitations and Future Research

The limitations of this work can be divided into two categories. Those that can be addressed through future research, and fundamental limitations that come from the scope of the work or physical constraints.

The exclusive use of synthetic data dominates the first category. As the trained model has never been validated using real camera footage of a runway, its actual performance under operational conditions remains unknown. The simulation-reality gap is a well-known problem in this field of research that arises from the use case itself, rather than being specifically related to the approach chosen in this work. It is practically impossible to purposefully collect real runway footage with placed objects for safety and regulatory reasons [3, 9]. Since collecting real runway footage remains practically infeasible, future work should address this gap through domain randomization, like varying lighting, textures, and environmental conditions across simulation runs, as well as domain adaptation techniques applied to the trained model [37, 38]. Another limitation that should be addressed is the training duration. With 10 epochs in Phase 1 and 50 epochs in Phase 2, the training curves suggest that the model has not fully reached a plateau yet. While this duration was sufficient to validate the effectiveness of the methodology, further training could improve the detection performance, particularly for underrepresented classes. Closely related to this is the ODD coverage. Even though the conditions are appropriate and well-chosen, the work focuses on approaches to RWY 23 under cloudy weather conditions and daylight between 10 a.m. and 2 p.m. Here, there is room for extensions using other runways at Hamburg airport or even other airport environments. Follow-up work should also include night operations, rain, or fog to achieve an operationally robust ODD. The dataset structure itself can be expanded as well. Each frame contains a maximum of one object and lacks multi-object scenarios. A key advantage of the simulation-based setup developed for this study is that extensions like these can be implemented with relatively little effort, as the existing pipeline can be reused directly. Since this thesis focuses on the development of a methodology and its implementation, these extensions would be well-suited for future work that focuses exclusively on the practical applications. Furthermore, this work exclusively uses the medium variant of YOLO11. Future work should evaluate the full range of model sizes to identify the optimal trade-off between detection performance and computational efficiency for deployment on airborne hardware. Additionally, newer YOLO versions released during the course of this work, such as YOLO26, should be considered in future studies, as they may offer improved detection capabilities for small and rare object classes [58]. Lastly, the synthetic dataset

has not been validated against existing FOD benchmarks, making it difficult to compare its performance with other systems.

Another important limitation concerns the model's behavior regarding small and rare classes. The dataset has a background frame rate of 18 %. As a result of this high background rate, the model has learned to avoid predictions when uncertain. This is clear in the classes *drone* and *suitcase*. They both have a precision of 1.0 in Phase 2, but a recall close to zero. The model rarely recognizes these objects, but when it does make a prediction, it is correct. For a safety-critical use case, where recall is more important than precision, this conservative behavior is problematic. This limitation can be addressed by reducing the background rate and increasing the number of positive training examples for these classes. It is also worth noting that *container* and *debris_pile* regressed from Phase 1 to Phase 2, suggesting that full fine-tuning did not benefit all classes equally and that the training process may require class-specific adjustments.

The second category includes limitations that cannot be overcome by collecting more data or using different training strategies. The six selected EASA objectives only cover a subset of the objectives and just three phases of the learning assurance process. For example, other phases like the model verification are deliberately outside the scope of this work and must be addressed in future research. The optical resolution of small objects and their size itself represent a fundamental limitation. At typical approach distances, the classes *drone*, *person*, and *suitcase* occupy only two to three pixels and are barely visible, even to the human eye. This limitation is not due to the model, but rather a physical consequence of camera position and object choice. It can only be overcome by using a multi-sensor setup with additional cameras installed on the runway. Finally, like all deep neural networks, YOLO11m offers only limited interpretability. Individual predictions cannot be directly traced back to rule-based parameters, which is relevant in the context of EASA requirements for transparency and traceability [1]. Nevertheless, for vision-based perception tasks of this complexity, deep neural networks still represent the appropriate tool, as no rule-based alternative offers comparable detection performance across diverse object classes and viewing conditions.

6.3 Contribution to Theory

This thesis addresses the identified gap between regulatory guidance and practical implementation by developing a Safety-by-Design methodology that translates six EASA objectives into concrete development steps.

In previous research, individual aspects of this gap were addressed. For instance, studies have examined the definition of ODDs [30, 31], the derivation

of data requirements from operational conditions [33], and the formulation of overarching engineering frameworks [12, 26]. What has been missing is a consistent, application-oriented methodology that translates these elements into a development process and validates this process against a concrete, safety-critical use case. The proposed Safety-by-Design methodology addresses this gap directly. Furthermore, the 27.8 % improvement from Phase 1 to Phase 2 supports the two-phase training strategy proposed by Lenhard et al. [59], providing empirical evidence for its effectiveness in domain-specific fine-tuning scenarios.

A second gap exists at the applicant level. Although there are annotated datasets for camera-based object detection in aviation contexts [46, 78], none address the specific combination of runway FOD detection in accordance with EASA data management. For instance, the LARD dataset focuses on runway recognition rather than objects on the runway [46]. Existing FOD detection systems are primarily designed for ground-based camera installations rather than from the perspective of a landing aircraft [3, 9]. The generated dataset that was developed in this thesis is the first to combine this specific approach perspective with an ODD-compliant data management process according to EASA objectives, including traceability, DQRs verification, and independent run-based partitioning. The finding that *drone* and *suitcase* are undetectable at approach distances confirms the assessment of Shan et al. that no single sensor type can meet all operational detection requirements [3].

A third theoretical contribution is the explicit positioning of the methodology within the W-shaped learning assurance process. By clearly specifying that the three phases, data management, learning process management, and model training, are covered and those excluded, the methodology creates a structure that future work could extend. Future work addressing other phases can build on the existing methodology without redesigning the upstream process. This makes the contribution transferable outside the specific FOD detection use case.

A fourth and last contribution is of an empirical nature. During the application of the methodology, two data quality problems were found, as discussed in Section 4.3.3. This gives direct evidence that the iterative quality assurance methodology is not merely theoretical, but detects real issues that would otherwise have stayed unnoticed until after training. Additionally, the results confirm the simulation-reality gap discussed in the related work chapter [36, 38]. The model achieves strong performance on synthetic test data, but has not been validated on real runway footage. This highlights that this limitation is not unique to this study but is a structural issue in the field.

6.4 Managerial Implications

For development teams, the methodology provides a structured set of tools that translate abstract EASA objectives into concrete work steps. The ODD-to-Data Traceability Matrix and the DQRs verification table are not only theoretical constructs, but also prove effective in detecting data quality issues at an early stage of the development process. As demonstrated in Section 4.3.3, two problems that would likely have gone unnoticed until after training were identified and corrected before any model training began. This demonstrates a direct increase in efficiency, reducing the risk of labor-intensive corrections later in the development cycle. The run-based dataset split also provides a reusable approach for keeping independence between training and test data in simulation-based projects. The two-stage training procedure, first stabilizing the detection head and then fine-tuning the full network, can be applied to other transfer learning projects. It can serve as a reusable template for use cases with limited domain-specific data.

FOD detection addresses a problem with substantial safety and financial implications. If indirect costs such as delays and cancellations are included, it is estimated that FOD costs the aviation industry up to $22.7 billion annually [7]. Several historical incidents, such as the crash of Air France Flight 4590, demonstrate that even small objects on the runway can have catastrophic outcomes [4]. Traditional manual runway inspection is labor-intensive and rarely performed [3], highlighting the urgent need for automated detection. In this context, the results provide airport operators with a clear and evidence-based understanding of the capabilities and limitations of AI-based FOD detection from an approaching aircraft. Distinct and large objects, such as aircraft or vehicles, can be detected with solid accuracy (mAP@0.5 of 0.547 and 0.417). The class *debris_scatter* has the largest physical size on the runway, achieving almost a perfect detection performance (mAP@0.5 of 0.954). Small objects such as drones or suitcases are not reliably detectable at typical approach distances, because they appear in only two to three pixels in the frames. This finding confirms the observation of Shan et al. that no single sensor type can meet all detection requirements [3]. Using only an onboard camera solution does not replace dedicated ground-based multi-sensor setups. However, not all airports operate ground-based FOD detection systems due to the significant infrastructure and maintenance costs involved [3]. In such cases, an onboard camera system mounted on the approaching aircraft could serve as a cost-effective complementary layer, providing additional detection coverage during the critical final approach phase without requiring runway-side infrastructure.

For certification authorities, this work provides a concrete example of what a development process could look like in the initial phases of the learning assurance

W-shaped process. Each development step can be tracked back to an EASA objective and is documented in structured elements. The ODD-to-Data Traceability Matrix establishes the link between operational conditions and data requirements (DM-01). The DQRs verification table documents quality checks and corrective actions (DM-03). The run-based split demonstrates independent data partitioning (DM-06). The model architecture definition documents the architectural choices and their justification (LM-01). The hyperparameter table and two-phase training specification fully describe the learning process before training begins (LM-02). The training logs provide complete documentation of the training process (LM-05). This level of traceability reflects the required transparency and reproducibility from EASA to establish confidence in data-driven systems [1].

6.5 Conclusion

This work started by addressing the question of how a Safety-by-Design methodology should look, based on selected EASA objectives, to develop a simulation-based dataset and an AI model for safety-critical object detection on a runway. The answer is a structured methodology that translates six of the EASA objectives into three development phases. This methodology was validated through the creation of a synthetic dataset and the training of a YOLO11m detection model. The results show that the regulatory requirements can be put into practice.

At the same time, the work is transparent about its limitations. The methodology covers only a subset; the model has been validated exclusively on synthetic data and does not cover all possible approach scenarios. Expanding the ODD coverage, validating the model on real approach footage, and addressing the remaining phases of the W-shaped learning assurance process are logical next steps towards a certifiable system. The methodology developed in this thesis provides a solid foundation on which future work can be built.

As in many other domains, the adoption of AI in aviation is increasing rapidly, with applications expected to grow by approximately 35 % per year [11]. As machine learning is increasingly applied to safety-critical tasks, developing these systems safely and making them certifiable becomes as important as advancing the technology itself. This thesis provides a practical answer to exactly this challenge by showing how abstract regulatory objectives can be translated into a structured development process and validated against a real safety-critical use case. This study makes a concrete and meaningful contribution by providing a structured, regulation-aligned methodology for developing safe and certifiable AI-based systems in aviation. This methodology makes such systems not only theoretically possible but also practically achievable.

A Selected EASA Objectives

Table 7: Selected objectives of EASA's Concept Paper that are considered for the methodology of this thesis [1].

Objective	**Description**
DM-01	"The applicant should identify data sources and collect data in accordance with the defined ODD, while ensuring satisfaction of the defined DQRs, in order to drive the selection of the training, validation and test data sets."
DM-03	"The applicant should define the data preparation operations to properly address the captured requirements (including DQRs)."
DM-06	"The applicant should distribute the data into three separate data sets which meet the specified DQRs in terms of independence (as per Objective DA-04): – the training data set and validation data set, used during the model training; – the test data set used during the learning process verification, and the inference model verification."
LM-01	"The applicant should describe the ML model architecture."
LM-02	"The applicant should capture the requirements pertaining to the learning management and training processes, including but not limited to: – model family and model selection; – learning algorithm(s) selection; – explainability capabilities of the selected model; – activation functions; – cost/loss function selection describing the link to the performance metrics; – model bias and variance metrics and acceptable levels (only in supervised learning); – model robustness and stability metrics and acceptable levels; – training environment (hardware and software) identification; – model parameters initialization strategy; – hyper-parameters and parameters identification and setting; – expected performance with training, validation, and test data sets."
LM-05	"The applicant should document the result of the model training."

B ODD-to-Data Traceability Matrix

Table 8: ODD-to-Data Traceability Matrix.

(1) ODD-ID	(2) ODD Parameter	(3) Constraint / Categories	(4) Data-rel.	(5) DR-ID	(6) Derived Data Requirement	(7) SRC-ID	(8) Source Type
ODD-01	Time of Day	Daytime; 10:00–14:00	Yes	DR-01	The dataset shall contain image data captured under daytime conditions between 10:00 and 14:00. Metadata indicating the time of capture shall be available.	SRC-01	Internal / Synthetic (Simulator)
ODD-02	Weather Condition	Cloudy weather; full cloud coverage; no rain	Yes	DR-02	All images shall reflect cloudy weather conditions with full cloud coverage and without precipitation.	SRC-01	Internal / Synthetic (Simulator)
ODD-03	Airport & Runway	Hamburg Airport (EDDH); single defined approach configuration	Yes	DR-03	All images shall depict the runway environment of Hamburg Airport on the defined approach configuration.	SRC-01	Internal / Synthetic (Simulator)
ODD-04	Ownship Trajectory	Start: 800 m before threshold, 138 m (453 ft) AGL; End: 400 m past threshold, 24 m (79 ft) AGL	Yes	DR-04	The dataset shall cover the full approach trajectory across the defined altitude and distance range. Trajectory parameters are derived from real-world operational data.	SRC-01, SRC-02	Internal / Synthetic (Simulator); Real-world operational data (HAM)
ODD-05	Ownship Speed	$66\,\mathrm{m\,s^{-1}}$ (128 kn)	No	—	The ownship speed does not influence the visual content of individual image frames, the labeling, or the model behavior, as no sequential modeling is performed.	—	—
ODD-06	Object Classes	Airplane, container, debris_scatter, debris_pile, drone, person, suitcase, vehicle	Yes	DR-05	The dataset shall contain annotated instances of all 8 object classes, placed at varying positions on or above the runway.	SRC-01	Internal / Synthetic (Simulator)
ODD-07	Camera System	1920 px × 1080 px, Pitch: −15°, FOV: 60°	Yes	DR-06	All generated images shall have a resolution of 1920 px × 1080 px. The frame rate of $30\,\mathrm{s^{-1}}$ shall be reflected in the temporal sampling of the simulation.	SRC-01	Internal / Synthetic (Simulator)

C Data Quality Requirements

Table 9: Data Quality Requirements (DQRs), verification methods, and evaluation results for the generated dataset.

DQR-ID	DR-ID	Requirement	Verification Method	Result
DQR-01	DR-01	All frames shall represent daytime conditions between 10:00 and 14:00.	Verify simulation time parameter	✓
DQR-02	DR-02	All frames shall show cloudy daytime conditions with full cloud coverage and without precipitation.	Verify weather setting in simulation config	✓
DQR-03	DR-03	All frames shall depict the Hamburg Airport RWY 23 environment.	Verify that a single, fixed scene is used	✓
DQR-04	DR-04	The dataset shall cover the full approach trajectory from −800 m to +400 m at a step size of 5 m (241 waypoints).	Verify waypoint range and step size in the dataset script	✓
DQR-05	DR-05	The dataset shall contain annotated instances of all 8 defined object classes.	Count annotated instances per class across all splits; threshold: $\geq$ 1 instance per class	✓*
DQR-06	DR-05	Class label IDs shall correctly map to the 8 target classes without ID collisions.	Compare COCO class IDs with local simulation class indices; threshold: 0 collisions	✓*
DQR-07	DR-05	The proportion of background frames (frames with no detectable object) shall not exceed 20 %.	Count frames with empty label files; threshold: $\leq$ 20 %	✓*
DQR-08	DR-06	All images shall have a resolution of 1920 px × 1080 px.	Verify image dimensions programmatically	✓

* Fulfilled after corrective action; see text for details.

D Contribution Statement

This master's thesis was prepared in collaboration with the German Aerospace Center (DLR), Institute for AI Safety and Security. The work was embedded in the research context of safety-by-design development for AI-based aviation systems. DLR supported it through access to technical resources, server-based computing infrastructure, domain expertise, and regular supervision. The author was responsible for the independent development and written elaboration of the thesis, including the literature review, the derivation of the proposed safety-by-design methodology, its application to the runway object detection use case, the setup of the required development environment, and the implementation and validation of the simulation-based dataset creation and AI model development workflow. Contributions from DLR consisted of research guidance, technical feedback, access to technical resources and high-performance server infrastructure, and support in defining the practical use case and its aviation-specific context.

References

[1] EASA, "Easa concept paper: Guidance for level 1 & 2 machine learning applications," European Union Aviation Safety Agency (EASA), Postfach 10 12 53, 50452 Cologne, Germany, techreport, 04 2024. [Online]. Available: https://www.easa.europa.eu/en/document-library/general-publications/easa-artificial-intelligence-concept-paper-issue-2

[2] Boeing, "Statistical summary of commercial jet airplane accidents," Boeing Commercial Airplanes, Seattle, Washington, USA, Tech. Rep., 07 2009. [Online]. Available: https://www.airsafe.com/events/models/statsum2008.pdf

[3] J. Shan, L. Miccinesi, A. Beni, L. Pagnini, A. Cioncolini, and M. Pieraccini, "A review of foreign object debris detection on airport runways: Sensors and algorithms," *Remote Sensing*, vol. 17, no. 2, p. 225, 2025, submission received: 18 December 2024; Revised: 4 January 2025; Accepted: 7 January 2025; Published: 9 January 2025. [Online]. Available: https://www.mdpi.com/2072-4292/17/2/225

[4] Bureau Enquêtes-Accidents (BEA), "Accident on 25 july 2000 at la patte d'oie in gonesse (95) to the concorde registered f-btsc operated by air france," Ministere De L'equipement Des Transports Et Du Logement – Bureau D'enquetes Et D'analyses Pour La Securite De L'aviation Civile, France, Tech. Rep., 07 2000. [Online]. Available: https://asn.flightsafety.org/asndb/323465

[5] Wall Street Journal. (2026) Dozens of close calls preceded deadly LaGuardia crash. [Online]. Available: https://www.wsj.com/business/airlines/dozens-of-close-calls-preceded-deadly-laguardia-crash-065abbf3

[6] T. Chauhan, C. Goyal, D. Kumari, and A. K. Thakur, "A review on foreign object debris/damage (fod) and its effects on aviation industry," *Materials Today: Proceedings*, vol. 33, no. Part 7, pp. 4336–4339, 2020.

[7] B. G. Hilburn and B. S. Pesmen, "Foreign object debris detection system cost-benefit analysis," Federal Aviation Administration, Tech. Rep. DOT/FAA/TC-22/47, 2023.

[8] M. Babitz. (2025, April) The silent airside threat: How AI is fighting foreign object debris (FOD) and revolutionizing airport operations. Airports Council International – North America. [Online]. Available: https://airportscouncil.org/2025/04/30/the-silent-airside-threat-how-ai-is-fighting-foreign-object-debris-fod-and-revolutionizing-air

[9] M. Noroozi and A. Shah, "Towards optimal foreign object debris detection in an airport environment," *Expert Systems With Applications*, vol. 213, p. 118829, 2023.

[10] A. Schoeman and A. Panday, "Certification of ai-based aviation systems: A methodology for continuous safety assurance across the system life cycle," in *Proceedings of the 2025 SAIMechE Central Branch Conference on Mechanical Engineering and Related Disciplines*, vol. 132, no. 1. Basel, Switzerland: MDPI, 2026, p. 7. [Online]. Available: https://doi.org/10.3390/engproc2026132007

[11] Precedence Research, "Artificial intelligence in aviation market size, share, and trends 2025 to 2034," Precedence Research, resreport 1748, 05 2022. [Online]. Available: https://www.precedenceresearch.com/artificial-intelligence-in-aviation-market

[12] J. M. Christensen, T. Stefani, A. Anilkumar Girija, E. Hoemann, A. Vogt, V. Werbilo, U. Durak, F. Köster, T. Krüger, and S. Hallerbach, "Formulating an engineering framework for future ai certification in aviation," *Aerospace*, vol. 12, no. 6, pp. 1–27, 05 2025. [Online]. Available: https://www.mdpi.com/2226-4310/12/6/482

[13] B. Luettig, Y. Akhiat, and Z. Daw, "Ml meets aerospace: Challenges of certifying airborne ai," *Frontiers in Aerospace Engineering*, vol. 3, p. 1475139, 2024.

[14] G. Vidot, C. Gabreau, I. Ober, and I. Ober, "Certification of embedded systems based on machine learning: A survey," *arXiv*, 2021. [Online]. Available: https://arxiv.org/abs/2106.07221

[15] European Organization for Civil Aviation Equipment (EUROCAE), "Software considerations in airborne systems and equipment certification," European Organization for Civil Aviation Equipment (EUROCAE), Malakoff, France, techreport ED-12C, 01 2012. [Online]. Available: https://eshop.eurocae.net/eurocae-documents-and-reports/ed-12c-with-corrigendum-1/

[16] RTCA, Inc., "Software considerations in airborne systems and equipment certification," GlobalSpec, Washington, DC, USA, techreport DO-178C, 01 2012.

[17] SAE International, "Guidelines for development of civil aircraft and systems (ARP4754A)," SAE International, Tech. Rep. ARP4754A, 12 2010. [Online]. Available: https://www.sae.org/standards/content/arp4754a/

[18] A. Sherifi, "Deep reinforcement learning based systems for safety critical applications in aerospace," *arXiv*, Dec. 2024. [Online]. Available: https://arxiv.org/abs/2412.16489

[19] A. M. Diaz and C. M. C. Llompart, "Particularities of certifying artificial intelligence in military aviation," in *Deutscher Luft- und Raumfahrtkongress 2024*. Germany: Deutsche Gesellschaft für Luft- und Raumfahrt (DGLR), 2024.

[20] S. Hallerbach, "Simulation-based testing of cooperative and automated vehicles," Ph.D. dissertation, Carl von Ossietzky Universität Oldenburg, 06 2020. [Online]. Available: https://oops.uni-oldenburg.de/id/eprint/4649

[21] H. Bello, D. Geißler, L. Ray, S. Müller-Divéky, P. Müller, S. Kittrell, M. Liu, B. Zhou, and P. Lukowicz, "Towards certifiable ai in aviation: Landscape, challenges, and opportunities," *arXiv*, 2024. [Online]. Available: https://arxiv.org/abs/2409.08666

[22] SAE International, "Artificial intelligence in aeronautical systems: Statement of concerns: Air6988," SAE International, Tech. Rep. AIR6988, 04 2021. [Online]. Available: https://www.sae.org/standards/content/air6988/

[23] European Union Aviation Safety Agency (EASA), "Easa concept paper: First usable guidance for level 1 machine learning applications," European Union Aviation Safety Agency (EASA), Postfach 10 12 53, 50452 Cologne, Germany, techreport, 04 2021.

[24] ——, "Artificial intelligence roadmap 2.0," European Union Aviation Safety Agency (EASA), Postfach 10 12 53, 50452 Cologne, Germany, techreport, 05 2023. [Online]. Available: https://www.easa.europa.eu/en/document-library/general-publications/easa-artificial-intelligence-roadmap-20

[25] J.-G. Durand, A. Dubois, and R. J. Moss, "Formal and practical elements for the certification of machine learning systems," in *2023 IEEE/AIAA 42nd Digital Avionics Systems Conference (DASC)*. Barcelona, Spain: IEEE, 2023, pp. 1–10.

[26] F. Werner, J. M. Christensen, T. Stefani, F. Köster, E. Hoemann, and S. Hallerbach, "Formulating a learning assurance-based framework for ai-based systems in aviation," *Aerospace*, 2026, german Aerospace Center (DLR).

[27] C. Torens, F. Jünger, S. Schirmer, P. Nagarajan, S. Schopferer, D. Zhukov, and J. Dauer, "Runtime monitoring of operational design domain to safeguard

machine learning components," *CEAS Aeronautical Journal*, vol. 16, pp. 973–991, 2025.

[28] ISO, *ISO 34503: Road Vehicles – Test scenarios for automated driving systems – Specification for operational design domain.* Berlin: Beuth Verlag, 08 2023.

[29] M. Hirschle, D. Kirov, R. Aievola, and J. Adamy, "Coverage-driven synthetic data generation for machine learning assurance," in *2024 AIAA DATC/IEEE 43rd Digital Avionics Systems Conference (DASC).* San Diego, CA, USA: IEEE, 09 2024, pp. 1–10.

[30] C. Torens, S. Gupta, N. Roy, J. Sprockhoff, and U. Durak, "From operational design domain to runtime monitoring of ai-based aviation systems," in *2024 AIAA DATC/IEEE 43rd Digital Avionics Systems Conference (DASC).* San Diego, CA, USA: IEEE, 09 2024, pp. 1–9.

[31] F. Werner, "Ensuring the trustworthy development of ai-based applications compatible to future easa regulations," mathesis, Ulm University of Applied Sciences, Ulm, Germany, 01 2025. [Online]. Available: https://elib.dlr.de/212205/

[32] G. Pedroza, M. Paquet, and B. Dion, "A digital engineering methodology for design, exploration and validation of safety-critical software for integrating ai-based algorithms," *INCOSE International Symposium*, vol. 35, no. 1, p. e70011, 2025.

[33] C. Cappi, N. Cohen, M. Ducoffe, C. Gabreau, L. Gardes, A. Gauffriau, J.-B. Ginestet, F. Mamalet, V. Mussot, C. Pagetti, and D. Vigouroux, "How to design a dataset compliant with an ml-based system odd?" in *12th European Congress on Embedded Real Time Software and Systems (ERTS)*, Toulouse, France, 06 2024, pp. 1–10. [Online]. Available: https://hal.science/hal-04614554

[34] T. Stefani, M. Jameel, I. Gerdes, R. Hunger, C. Bruder, E. Hoemann, J. M. Christensen, A. A. Girija, F. Köster, T. Krüger, and S. Hallerbach, "Towards an operational design domain for safe human-ai teaming in the field of ai-based air traffic controller operations," in *2024 AIAA DATC/IEEE 43rd Digital Avionics Systems Conference (DASC).* San Diego, CA, USA: IEEE, 09 2024, pp. 1–10.

[35] G. Pedroza and B. Dion, "Design, exploration and evaluation of safety-critical software for integrating ai/ml-based algorithms," in *embedded world*

Conference 2025. Nuremberg, Germany: embedded world Exhibition & Conference, 2025, conference paper.

[36] S. Gupta, U. Durak, O. Ellis, and C. Torens, "From operational scenarios to synthetic data: Simulation-based data generation for ai-based airborne systems," in *AIAA Scitech Forum*, 2022, based on EASA guidance for machine learning applications.

[37] J. Rüter, T. Maienschein, S. Schirmer, S. Schopferer, and C. Torens, "Filling the gaps: Using synthetic low-altitude aerial images to increase operational design domain coverage," *Sensors*, vol. 24, no. 4, p. 1144, 02 2024. [Online]. Available: https://www.mdpi.com/1424-8220/24/4/1144

[38] T. R. Dieter, A. Weinmann, S. Jäger, and E. Brucherseifer, "Quantifying the simulation–reality gap for deep learning-based drone detection," *Electronics*, vol. 12, no. 10, p. 2197, 2023.

[39] S. R. Gattnar, H. Späth, Y. Akhiat, and Z. Daw, "From odd to data: An end-to-end toolchain for synthetic data generation – a case study on ai-based runway detection," in *2025 AIAA DATC/IEEE 44th Digital Avionics Systems Conference (DASC)*. IEEE, 2025.

[40] D. J. Fremont, E. Kim, T. Dreossi, S. Ghosh, X. Yue, A. L. Sangiovanni-Vincentelli, and S. A. Seshia, "Scenic: a language for scenario specification and data generation," *Machine Learning*, vol. 112, no. 10, pp. 3805–3849, 02 2022.

[41] T. Dieter, A. Weinmann, and E. Brucherseifer, "Generating synthetic data for deep learning-based drone detection," in *AIP Conference Proceedings*, vol. 2939. AIP Publishing, 2023, p. 030007.

[42] W. Dai, Z. Zhai, D. Wang, Z. Zu, S. Shen, X. Lv, S. Lu, and L. Wang, "Yomo-runwaynet: A lightweight fixed-wing aircraft runway detection algorithm combining yolo and mobilerunwaynet," *Drones*, vol. 8, no. 7, p. 330, 2024.

[43] P. Thai, S. Alam, N. Lilith, and B. T. Nguyen, "A computer vision framework using convolutional neural networks for airport-airside surveillance," *Transportation Research Part C: Emerging Technologies*, vol. 137, p. 103590, 2022.

[44] M. Farhadmanesh, A. Rashidi, P. Schonfeld, J. Rakas, and N. Marković, "Aircraft surface movement and operation monitoring systems in general aviation and commercial airports: A state-of-the-art review," *Iranian Journal*

of Science and Technology, Transactions of Civil Engineering, vol. 49, pp. 1009–1030, 2025.

[45] I. Feitosa, B. Santos, and P. G. Almeida, "Automated and intelligent inspection of airport pavements: A systematic review of methods, accuracy and validation challenges," *Future Transportation*, vol. 5, no. 4, p. 183, 2025.

[46] M. Ducoffe, M. Carrere, L. Féliers, A. Gauffriau, V. Mussot, C. Pagetti, and T. Sammour, "LARD - landing approach runway detection - dataset for vision based landing," 04 2023, working paper or preprint. [Online]. Available: https://hal.science/hal-04056760

[47] Y. Bougacha, G. Delhomme, M. Ducoffe, A. Fuchs, J.-B. Ginestet, J. Girard, S. Kraïem, F. Mamalet, V. Mussot, C. Pagetti, and T. Sammour, "LARD 2.0: Enhanced dataset and benchmarking for autonomous landing systems," 2026, european Congress of Embedded Real Time Systems (ERTS), 2026. [Online]. Available: https://arxiv.org/abs/2603.26748

[48] S. S. A. Zaidi, M. S. Ansari, A. Aslam, N. Kanwal, M. Asghar, and B. Lee, "A survey of modern deep learning based object detection models," *Digital Signal Processing*, vol. 126, p. 103514, 2022.

[49] T. Diwan, G. Anirudh, and J. V. Tembhurne, "Object detection using yolo: challenges, architectural successors, datasets and applications," *Multimedia Tools and Applications*, vol. 82, pp. 9243–9275, 2023.

[50] S. Y. Mohammed, "Architecture review: Two-stage and one-stage object detection," *Franklin Open*, vol. 12, p. 100322, 2025.

[51] M. Kotthapalli, D. Ravipati, and R. Bhatia, "YOLOv1 to YOLOv11: A comprehensive survey of real-time object detection innovations and challenges," 2025.

[52] T. Shehzadi, K. A. Hashmi, M. Liwicki, D. Stricker, and M. Z. Afzal, "Object detection with transformers: A review," *Sensors*, vol. 25, no. 19, p. 6025, 2025.

[53] Z. Wang, "Deep learning-based foreign object detection method for aviation runways," *Applied Mathematics and Nonlinear Sciences*, vol. 8, no. 1, pp. 3187–3202, 2023.

[54] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE Conference*

on Computer Vision and Pattern Recognition (CVPR). IEEE, 2016, pp. 779–788.

[55] A. Vijayakumar and S. Vairavasundaram, "YOLO-based object detection models: A review and its applications," *Multimedia Tools and Applications*, vol. 83, pp. 83 535–83 574, 2024.

[56] P. Li and H. Li, "Research on fod detection for airport runway based on yolov3," in *Proceedings of the 39th Chinese Control Conference*, Shenyang, China, July 2020, pp. 7096–7099.

[57] Z. Qi, D. Ma, J. Xu, A. Xiang, and H. Qu, "Improved yolov5 based on the attention mechanism and fasternet for foreign object detection on railway and airway tracks," in *2024 Asian Conference on Communication and Networks (ASIANComNet)*. IEEE, 2024.

[58] R. Khanam and M. Hussain, "YOLOv11: An overview of the key architectural enhancements," 2024.

[59] T. R. Lenhard, A. Weinmann, S. Jäger, and E. Brucherseifer, "Deploying a feedback loop-based training strategy for deep learning-based drone detection," in *AIP Conference Proceedings*, vol. 3182, 2025, p. 060004.

[60] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollár, "Microsoft coco: Common objects in context," *arXiv*, 05 2014.

[61] T. R. Lenhard, A. Weinmann, and T. Koch, "Performance optimization of yolo-feder fusionnet for robust drone detection in visually complex environments," *arXiv preprint arXiv:2509.14012*, 2025.

[62] Y. Li, Y. Xia, G. Zheng, X. Guo, and Q. Li, "Yolo-rwy: A novel runway detection model for vision-based autonomous landing of fixed-wing unmanned aerial vehicles," *Drones*, vol. 8, no. 10, p. 571, 2024.

[63] T. Stefani, J. M. Christensen, A. Anilkumar Girija, S. Gupta, U. Durak, F. Köster, T. Krüger, and S. Hallerbach, "Automated scenario generation from operational design domain model for testing ai-based systems in aviation," *CEAS Aeronautical Journal*, vol. 16, no. 1, pp. 197–212, 11 2024.

[64] H. Bichri, A. Chergui, and M. Hain, "Investigating the impact of train/test split ratio on the performance of pre-trained models with custom datasets," *International Journal of Advanced Computer Science and Applications*, vol. 15, no. 2, pp. 331–339, 2024.

[65] J. Terven, D.-M. Cordova-Esparza, J.-A. Romero-González, A. Ramírez-Pedraza, and E. A. Chávez-Urbiola, "A comprehensive survey of loss functions and metrics in deep learning," *Artificial Intelligence Review*, vol. 58, p. 195, 2025.

[66] F. Karl, T. Pielok, J. Moosbauer, F. Pfisterer, S. Coors, M. Binder, L. Schneider, J. Thomas, J. Richter, M. Lang, E. C. Garrido-Merchán, J. Branke, and B. Bischl, "Multi-objective hyperparameter optimization in machine learning—an overview," *ACM Transactions on Evolutionary Learning*, vol. 3, no. 4, pp. 1–50, 2023.

[67] T. Song, "Comparative analysis of activation functions in simple convolutional neural networks," in *2024 IEEE 6th International Conference on Civil Aviation Safety and Information Technology (ICCASIT)*. IEEE, 2024, pp. 1031–1036.

[68] Q. Wang, Y. Ma, K. Zhao, and Y. Tian, "A comprehensive survey of loss functions in machine learning," *Annals of Data Science*, vol. 9, pp. 187–212, 2022.

[69] B. Bischl, M. Binder, M. Lang, T. Pielok, J. Richter, S. Coors, J. Thomas, T. Ullmann, M. Becker, A.-L. Boulesteix, D. Deng, and M. Lindauer, "Hyperparameter optimization: Foundations, algorithms, best practices, and open challenges," *WIREs Data Mining and Knowledge Discovery*, vol. 13, no. 2, p. e1484, 2023.

[70] T. Yu and H. Zhu, "Hyper-parameter optimization: A review of algorithms and applications," *arXiv preprint arXiv:2003.05689*, 2020.

[71] Monica and P. Agrawal, "A survey on hyperparameter optimization of machine learning models," in *2024 2nd International Conference on Disruptive Technologies (ICDT)*. IEEE, 2024, pp. 11–15.

[72] Epic Games. (2023) Unreal engine 5.3 is now available. [Online]. Available: https://www.unrealengine.com/blog/unreal-engine-5-3-is-now-available

[73] Microsoft Research. (2026) Projectairsim. [Online]. Available: https://github.com/iamaisim/ProjectAirSim

[74] W. Hematulin, P. Kamsing, T. Phisannupawong, T. Panyalert, S. Manuthasna, P. Torteeka, and P. Boonsrimuang, "Generating large-scale datasets for spacecraft pose estimation via a high-resolution synthetic image renderer," *Aerospace*, vol. 12, no. 4, p. 334, 04 2025. [Online]. Available: https://www.mdpi.com/2226-4310/12/4/334

[75] M. Schäfer, V. Lenders, I. Martinovic, and M. Wilhelm, "Bringing up opensky: A large-scale ads-b sensor network for research," in *Proceedings of the 13th International Conference on Information Processing in Sensor Networks (IPSN)*. Berlin, Germany: IEEE, April 2014, pp. 83–94.

[76] Z. He, Y. He, and Y. Lv, "Dt-yolo: An improved object detection algorithm for key components of aircraft and staff in airport scenes based on yolov5," *Sensors*, vol. 25, no. 6, p. 1705, 2025.

[77] M. Karl, "Optimized wide area media transport strategies," phdthesis, Universität des Saarlandes, 2015.

[78] T. R. Lenhard, A. Weinmann, K. Franke, and T. Koch, "Syndronevision: A synthetic dataset for image-based drone detection," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2025, pp. 7637–7647.