University of Padova

Department of Mathematics "Tullio Levi-Civita"

BRAIN, MIND AND COMPUTER SCIENCE

Curriculum in Computer Science and Innovation for Societal Challenges

XXXVI series

A TinyML-enabled approach to embed Machine Learning in Avionics Control Systems

Supervisor
Tullio Vardanega
University of Padova

Co-supervisor
Anna Spagnolli

Ph.D. Candidate
Zain Iqbal

Academic Year
2021-2023

This Ph.D. thesis is dedicated to my family:
My Mother ***Mussrat Iqbal***;
My Sister ***Aden Iqbal***;
My Brother ***Humza Iqbal***.

Acknowledgments

Embarking on this Ph.D. journey has transformed my life in unimaginable ways, a feat I could never have achieved without the incredible support and guidance from those around me.

First and foremost, I extend my deepest thanks to my supervisor, *Dr. Tullio Vardanega*, for his outstanding mentorship, insightful feedback, unwavering support and sincere concern throughout my academic endeavors. There were moments when I knew I must have been a source of worry, perhaps even a bit of a pain, with my incessant questions, doubts. Despite this, his motivation never waned. He believed in me even when I found it hard to believe in myself, tirelessly sourcing the best resources and opportunities to enrich my research and broaden my horizons. He was there, responding to my emails on weekends, carefully examining and revising my documents, and consistently encouraging me to excel. While his standards were demanding, his commitment to support me in achieving them was equally strong. His faith in my abilities was a consistent motivation that inspired me to strive for improvement, explore more deeply and strive for greater accomplishments.

The second very important phase of this venture continued in Germany, where I spent four months. This period was marked by my engagement with the Institut für Luftfahrtsysteme (ILS) at the University of Stuttgart. It is with profound respect and gratitude that I acknowledge *Dr. Zamira Daw* instrumental guidance and insightful suggestions during this phase. Her inspiration was unparalleled; not only did she serve as an exceptional mentor, but also as a beacon of strength and resilience as a strong woman in the field. Additionally, I extend my heartfelt thanks to *Dr. Umut Durak* for his hospitality and support at Clausthal University of Technology, Germany, making my stay both productive and memorable.

I would like to thank the reviewers, *Dr. Enrico Vicario* and *Dr. Zamira Daw* for providing their valuable feedback that helped me revise the thesis.

I extend my deepest gratitude to my family, whose continuous support and love have been my constant source of strength throughout this journey. To my mother, *Mussrat Iqbal*, for her endless sacrifices and encouragement, which have been the backbone of my achievements. To my sister, *Aden Iqbal*, whose companionship and understanding have lightened many of my days. To my brother, *Humza Iqbal*, for his continuous support, and to my sister-in-law, *Fizah Dogar*, for her kindness and care. My journey would not have been the same without the joy brought into my life by my three little munchkins: my nephews, *Zaviyar* and *Azlaan*, and my niece, *Elaya*. Their innocence and laughter have been a refreshing escape from the rigors of academic pursuit. My heart is full of love and gratitude for each of you for being my pillar of strength and for believing in me unconditionally.

I would also like to express my gratitude to my colleague, my friend and my partner *Enrico Cancelli*, whose love and support over the past year have been foundational. He has been my rock, consistently by my side through all the highs and lows, unwavering, and present when I needed him the most. His support and encouragement have given me the strength to face and overcome any challenge that came my way.

Furthermore,i am profoundly thankful to Enrico's family, *Marisa Ghitti*, *Lorenzo Cancelli*, and *Anna Maria Cancelli*. Their warm embrace, especially during my stay abroad in Italy, has meant the world to me. Their love and support have been a source of immeasurable comfort and strength. *Marisa*, especially, has moved beyond her supportive role to become the family I longed for, enveloping me with affection and generosity that has profoundly touched my heart and will be treasured forever.

Last but certainly not least, I extend my gratitude to all of my friends and colleagues. My deep gratitude goes out to each of you for providing me with strength and playing a pivotal role in this exciting journey. Your support and encouragement have been invaluable and I am truly grateful for your contribution to making this experience both rewarding and memorable.

ZAIN IQBAL
April 4, 2024

Abstract

Integrating Machine Learning (ML) into aviation's control systems heralds a significant leap forward, opening new pathways for increasing operational efficiency, enabling predictive maintenance, and refining decision-making processes in real time. This dissertation explores the innovative features of TinyML, a branch of machine learning created to operate with optimal efficiency in environments with limited resources, and its transformative impact on aviation control systems. It meticulously examines the challenges and strategies of embedding ML models within aviation's stringent frameworks, with a particular focus on managing Out-of-Distribution (OOD) instances that pose threats to system safety and reliability.

The incorporation of machine learning and embedded machine learning into avionics represents a transition towards making decisions based on data, enhancing the effectiveness, safety, and dependability of flight activities. The critical role these systems play in aviation underscores the importance of ensuring their dependability and performance, particularly their resilience against unexpected scenarios to safeguard aircraft and occupants alike.

Therefore, the integration strategy for ML in avionics encompasses not only the deployment of sophisticated algorithms, but also a comprehensive assurance framework to guarantee consistent and reliable performance under all operational conditions. This strategy combines theoretical advances in ML with practical considerations in the design of embedded systems to increase the overall safety and operational efficiency of aviation.

A cornerstone of this research is the formulation of an innovative integration strategy for ML systems, designed expressly for scenarios characterized by limited computational resources and a high demand for energy conservation. The research introduces two innovative multilayer early exit techniques in Deep Neural Networks (DNNs) with the goal of quickly and precisely identifying out-of-distribution (OOD) data in real-world scenario's by providing runtime assurance in embedded ML models. By integrating a detector of minimal complexity following each DNN layer, the proposed model facilitates an immediate cessation of inference upon the detection of OOD inputs, thereby elevating the system's computational efficiency and durability within stringent operational constraints.

Furthermore, this thesis conducts an exhaustive examination of sophisticated software testing methodologies and strategies applicable to ML-driven systems, underscoring the critical need to customize these technologies to align with the specialized requirements of safety-critical aviation environments. Achieving a harmonious balance between model dimensions, accuracy, and functionality, the research contributes significantly to the field by delivering solutions that are both practical and reliable for the operational breadth of these systems. The results underscore the superiority of the proposed model over existing alternatives, particularly in terms of computational efficiency and the detection of OOD data, as gauged by AUROC.

Additionally, the research expands on the critical role of Operational Design Domain (ODD) in conjunction with OOD. ODD defines the specific conditions under which a system is designed to operate, including environmental, geographical, and temporal constraints. Understanding ODD is crucial for accurately identifying OOD instances, data or situations that fall outside the system's designed operational parameters. By closely aligning OOD detection mechanisms with the defined ODD, the research highlights how ML models can more effectively anticipate and mitigate risks, ensuring higher safety and reliability in aviation systems. This nuanced exploration sets the foundation for subsequent research and the evolution of ML technologies within safety-critical settings, guiding toward smarter, more secure, and efficient aviation operations.

Contents

Listing of figures

Listing of tables

1 Introduction

In this chapter, we present the overarching concept of our doctoral research study, structured as follows.

- In the first section, we discuss the role of assurance for Machine Learning based System (MLSs) in safety-critical domain.
- Then, in the second section, we highlight our motivation to choose this specific topic for our study.
- The next two sections are dedicated to outlining the research gaps that we have identified and the research questions that we aim to address.
- In conclusion of this chapter, we offer a summary of our research contribution and key discoveries, along with an outline of the thesis structure.

1.1 Machine Learning based Systems (MLSs)

Machine Learning (ML) has long held a fascination for humanity, intriguing minds with the prospect of replicating intelligence within machines. This fascination can be traced back to as early as 1872, with Samuel Butler's envisioning of conscious machines in his novel "Erewhon."

The concept gained substantial momentum in the mid-20th century with Alan Turing's seminal work. In 1950, Turing proposed a test, famously known as the Turing Test [10], to evaluate a machine's ability to exhibit intelligent behavior equivalent to, or indistinguishable from, that of a human. This test marked a pivotal moment in the history of artificial intelligence and machine learning, setting a benchmark that would guide and inspire subsequent generations of research in the quest to achieve machine-based intelligence. ML, born from pattern recognition and the concept that computers can autonomously learn without being programmed for specific tasks, focuses on "optimizing a performance criterion using example data and past experience," as described by Alpaydin [11]. Unlike traditional software development, where behaviors are explicitly coded, ML thrives on processing complex high-dimensional data, discovering unknown patterns, and performing tasks automatically through training examples[12], with applications in image and audio processing, and natural language processing. The advent of ML and subsequent technological advances have greatly advanced these capabilities, leading to the development of sophisticated ML architectures, fueled by the abundance of data and the increased capabilities of graphical processing units. These advancements allow ML-based systems to sometimes outperform both traditional software and human capabilities. In contemporary practice, the integration of one or more ML components into software systems, referred to as Machine Learning based systems (MLSs), has become increasingly common.A system in which at least one of the components rely on machine learning techniques. In sectors such as healthcare [13], automotive [14], and manufacturing [15], systems often have high degrees of autonomy and safety, which causes them to be categorized as "safety-critical" (or, equivalently, "high assurance").

Safety-critical refers to systems or processes where failure or malfunction could result in serious consequences, such as loss of life, significant environmental harm, or substantial material damage. These systems are often found in industries where safety is paramount, including aviation, healthcare, nuclear power, and automotive sectors. The term underscores the critical nature of these systems in safeguarding against potential risks and hazards. In the realm of safety-critical systems, assurance refers to the systematic approach and methodology employed to guarantee the reliable and secure operation of these systems. This encompasses thorough testing, validation, verification, and ongoing monitoring to ensure that all components and functionalities adhere to the rigorous safety standards mandated for such systems.The relationship between safety-critical and assurance is intrinsic; assurance provides the necessary evidence and confidence that safety-critical systems will perform as intended, even in extreme or unexpected conditions. This includes ensuring that the system is free from defects, vulnerabilities

are addressed, and potential failures are anticipated and mitigated. In essence, assurance is the backbone of safety-critical systems, ensuring their reliability and protecting against the severe consequences of system failures.

1.1.1 Assurance of Machine Learning based Systems

Assurance

In machine learning, assurance is a comprehensive term that focuses on generating a well-founded belief that specific systems adhere to set criteria or fulfill predetermined claims, particularly within contexts where safety is paramount. The essence of the assurance process in machine learning lies in producing and compiling evidence, alongside constructing logical arguments, to affirm that a system operates securely and as anticipated across a variety of scenarios. Such evidence typically integrates both qualitative and quantitative aspects, showcasing the system's attributes related to safety and its competency in managing unexpected inputs or other forms of uncertainties.

Addressing the inherent uncertainties present in machine learning systems constitutes a core component of assurance. These uncertainties, stemming from the operational environment's complexity and the unpredictability of machine learning models, pose significant challenges to ensuring the system's safety and efficacy. Assurance activities are therefore geared towards identifying, measuring, and alleviating these uncertainties to bolster the system's dependability [16].

For machine learning applications, particularly within cyber-physical systems, quality assurance adopts an all-encompassing strategy that spans the planning and installation, operation, and data processing stages. This strategy is marked by a series of crucial steps including the delineation of development goals, the conceptualization of the machine learning application, the design of the measurement system, and its incorporation into the operational setting. Each phase plays a pivotal role in guaranteeing data integrity, which subsequently influences the analytical quality and the overall effectiveness of the machine learning application [17].

To encapsulate, assurance in machine learning is predicated on establishing a solid foundation for the system's safety and functionality, navigating through the intricacies and uncertainties involved. This process entails a systematic approach to gathering evidence, crafting arguments, and persistently evaluating the system's performance relative to established benchmarks.

Role of Assurance in MLSs

An ML-based system (MLSs in the sequel) is a system in which at least one part of its operation is dependent on (or constituted by) ML components.ML models are outnumbered by various software components in real-world MLSs., as required for human-to-machine and machine-to-machine interconnection, functional logic, and input transformations that cannot be inferred from the training data, as shown in Figure1.1.These systems utilize tensor algorithms to meticulously analyze data, employing ML models to autonomously make intelligent decisions. This process is fundamentally driven by the discovery of correlations, patterns, and knowledge extracted from training data, showcasing the sophisticated capabilities of ML in processing and interpreting vast datasets. However, It is worth mentioning that these systems do not depend solely on ML models. They encompass a multitude of traditional software components that work in concert to perform their functions effectively. These components may include data preprocessing modules, user interfaces, security protocols, and several other components that are necessary for the complete operation of the system, highlighting the complexity and multifaceted nature of such advanced software solutions. ML components differ vastly, for making and working, from traditional ones. ML models are computed using complex, nonlinear transformations, which depart from the inductive development style with which development stakeholders are usually familiar, except of course for the ML experts themselves. Such stakeholders thus need ways to gage the risks that the use of MLSs may incur and select the appropriate strategies for their assurance [18]. Owing to such a differentiating trait, MLSs belong in the class of software systems that currently lack a sufficiently robust test oracle, and which some researchers dub as "non-testable" [19], where traditional review, measurement, and quality assessment techniques are ineffective [20].

The appeal of adopting ML in safety-critical application domains is rising steadily in anticipation of the "intelligence bonus" expected from operational deployment of ML-boosted applications. Obviously, this drive has also caused much attention to the question of the level of assurance that can be achieved for such systems after they incorporate such additional components. Ways to reap the anticipated potential of ML and, more generally, of Artificial Intelligence (AI), are being explored also in materials science, robotics, and numerous other engineering systems. Application domains that are rich of field data (as in manufacturing, testing, and service) and that pursue multi-objective optimizations are especially suited for earning boosting from ML/AI data-driven algorithms. Owing to the computationally and storage intensive nature of current state-of-the-art implementations of ML, all data acquired from the field are

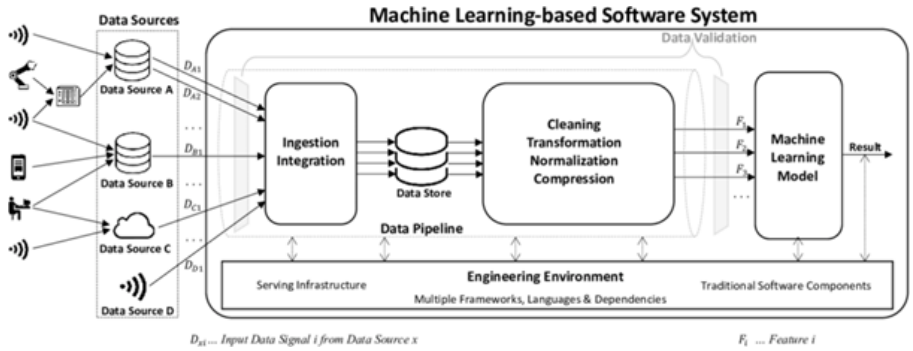

Figure 1.1: Machine Learning-based Software Systems [1] apply tensor algorithms to data and use ML models for making intelligent decisions automatically based on discovered correlations, patterns and knowledge inferred from training data. However, such systems include numerous other traditional software components to perform their duty.

transferred to the cloud for processing, which obviously incurs massive latency that renders the objective untenable for latency-sensitive embedded applications. Humans and ML-guided computers will need to interact in a safe, efficient, and effective manner even in a completely autonomous system scenario, which shall have to contemplate failure situations and assure awareness of human activity. For ML-based systems (MLSs), quality indicators include performance measures such as hit and false alarm rates, as well as an assessment of the goodness (fitness for purpose) of the data and methods used to train and test the system. MLSs have a data-driven algorithmic behavior that radically differs from the control-driven nature of their traditional predecessors, upon which the software verification body of knowledge rests. This particular trait of them puts the effectiveness of traditional verification approaches applied to MLSs, including testing, into question, which, in turn, causes serious concern to those considering use in high-assurance application domains.

The integration of ML models into the command-and-control loops of safety-critical systems represents a notable change in the architectural design of these embedded systems. In such configurations, ML models are not just ancillary components; they become integral to the core functioning of the system. These models are stored directly in the system's memory, ensuring immediate accessibility and rapid data processing. This proximity is crucial for real-time applications where even slight delays in response times can have significant consequences.

By embedding ML models within the system architecture, they operate on the same processor that runs the primary control functions of the system. This setup allows for seamless interaction between the ML models and other system components, facilitating efficient and timely decision-making. The advantage of this approach, as opposed to relying on external servers or cloud-based solutions, lies in its ability to provide rapid, on-the-spot processing ca-

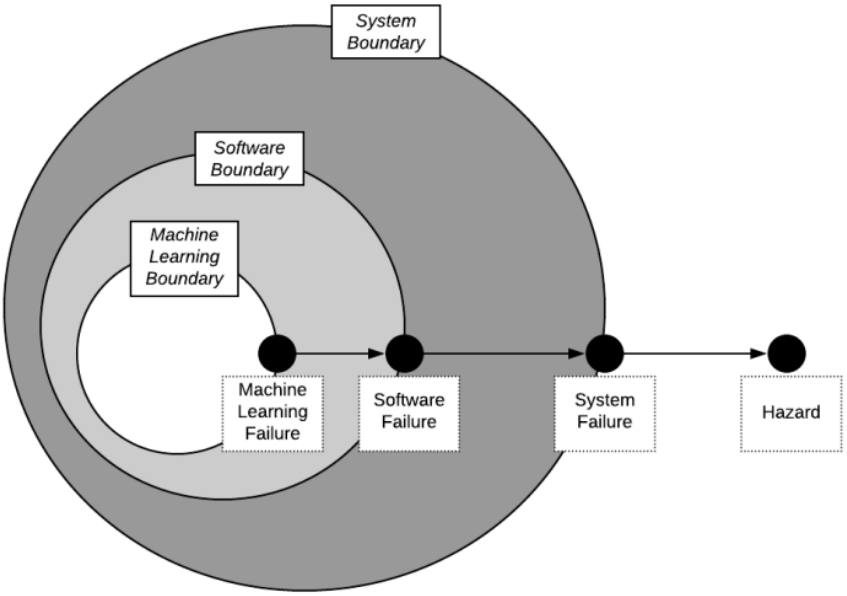

Figure 1.2: Depiction of a Simplified Sequence of Failure Events in Machine Learning Systems (MLSs): This illustration outlines the step-by-step progression of potential failure incidents within MLSs [2].

pabilities. This is essential in scenarios where decisions need to be made in real-time, such as in autonomous vehicles, medical devices, or avionic systems.

Furthermore, having ML models run on the system processor mitigates risks associated with network latency and connectivity issues that are inherent in configurations relying on external data processing. This ensures that the ML models can function effectively and reliably, even in environments where network connectivity is limited or unavailable. In many instances, ML-based systems are integral components of larger, more complex systems, and their malfunction can have cascading effects. For instance, a failure in an ML algorithm responsible for predictive maintenance in industrial settings can lead to unexpected equipment breakdowns, impacting overall operational efficiency and safety as shown in Figure 1.2. In sectors such as healthcare, an error in an ML-driven diagnostic tool could lead to misdiagnosis, affecting patient treatment plans and outcomes. Similarly, in the realm of cybersecurity, failure in ML-based threat detection systems can leave networks vulnerable to attacks. These dependencies underscore the necessity for robust and reliable ML-based systems, as their failure can compromise the functionality, safety, and integrity of the broader systems they support. Recognizing and mitigating these dependencies is crucial in system design and risk management strategies to ensure continuity and resilience in the face of ML-based system failures.

1.1.2 Role of TinyML in Assurance of MLSs

TinyML, which stands for Tiny Machine Learning, is an emerging effort that integrates ML into embedded devices of the Internet of Things (IoT) with the potential to revolutionize the aerospace industry and several other application domains with similar needs and characteristics [21]. which plays a crucial role in ensuring the reliability of Machine Learning (ML) based systems, especially in contexts where computing resources, power, and space are limited [22]. By enabling ML algorithms to run efficiently on small, low-power microcontrollers, TinyML facilitates real-time on-device data processing, this means it allows devices with limited processing power and energy resources, such as sensors and wearable devices, to perform data analysis locally, without needing to send data to the cloud for processing [23]. This local processing capability is crucial for applications where quick decision-making is essential, enabling devices to respond immediately to changes in data, enhance operational efficiency, and improve user experience by reducing latency, such as wearable health monitors or industrial sensors[24].Which can be crucial to ensuring uninterrupted functionality even in situations with limited or no connectivity.
TinyML combines hardware and software fit for resource scarce computation with the aim to enable ML models (particularly Deep Learning algorithms) on compact, relatively cheap, and power-efficient devices. Verifying function and performance assurances and confirming that the system meets certification and legal standards are all part of the testing process for a new product [25]. Additionally, TinyML can contribute to enhanced security and privacy, as data can be processed locally without needing to be transmitted to a central server. This aspect is particularly important in applications that handle sensitive information. Embedding ML in the way of the TinyML paradigm may aid the transition from traditional high-end systems to low-end clients as shown in Figure 1.3. TinyML offers a unique approach for the validation and assurance of machine learning models in constrained environments, focusing on the deployment of ML models on low-energy devices for edge case testing and real-world scenario simulations. The energy efficiency of TinyML models represents a pivotal advancement for battery-operated or remotely deployed systems, allowing these devices to operate for extended periods without significant power requirements, ideal for remote monitoring and IoT applications. Furthermore, by processing data locally, TinyML significantly improves data privacy and security, reducing the risk of data breaches during transmission [26].

The challenge of scalability in ML applications is addressed by the capability of TinyML to enable processing across devices of varying computational power, broadening the application

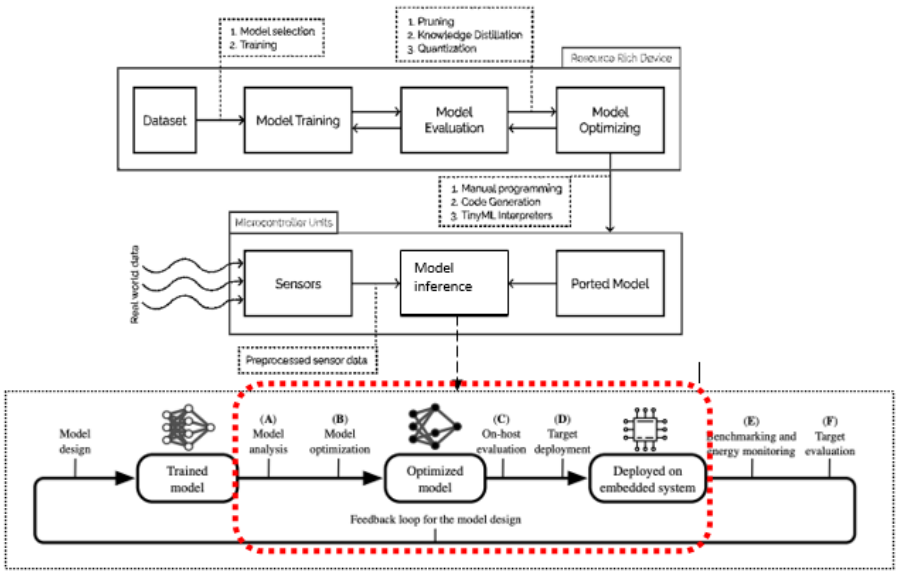

Figure 1.3: From [3] and [4], a modified workflow of Embedded MLSs includes portable benchmarking tools to quantify, analyse and optimize ML by deploying them directly on MCUs.

spectrum to include wearables and industrial sensors, thus leveraging the full potential of ML technology[27].

In avionics, the incorporation of TinyML marks a significant technological leap, embedding efficient ML algorithms into compact hardware to meet the demanding standards of aviation systems. Research by Thompson and Lee [28] underscores TinyML's role in facilitating real-time, onboard data processing and decision making, essential for navigation and system maintenance. Similarly, Patel et al.[29] emphasize the enhancement of system safety through TinyML, enabling continuous monitoring and quick response to anomalies, crucial for Out-of-Distribution (OOD) detection. This ensures that ML models remain effective and reliable in aviation's dynamic environment.

1.2 V-Model

The V-model, an adaptation of the traditional software development life cycle as shown in Figure 1.4, emphasizes a structured and systematic approach to both development and testing, making it especially suited for projects where safety and reliability are crucial, such as in the avionics, automotive, and medical device industries. At the foundation of the V-model is the Requirements Analysis phase, where system requirements are meticulously defined. Following

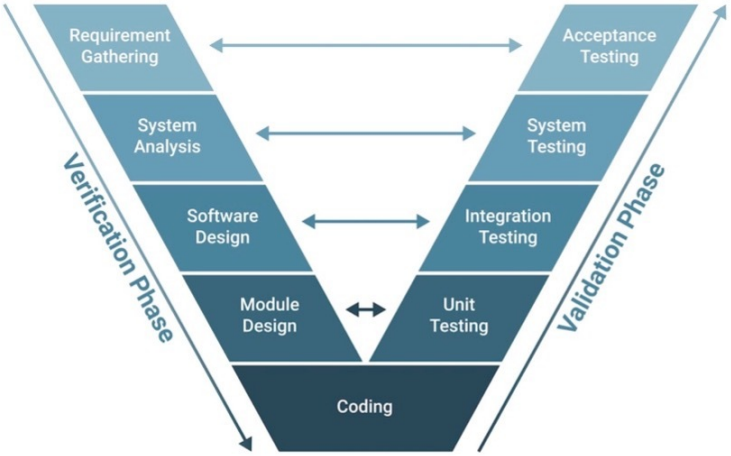

Figure 1.4: Diagram of the V Model in Software Development: Detailing Sequential Phases from Requirements Specification through Coding and Back up through Incremental Testing Levels

this, the System Design phase outlines the system's architecture, which is then broken down into smaller units or modules during the Architectural Design phase. These modules receive detailed planning in the Module Design phase, leading into the Implementation phase, where the actual coding takes place.

As the development process progresses from implementation, testing phases begin with Unit Testing to assess each module independently, followed by Integration Testing to ensure modules work seamlessly together. The System Testing phase then evaluates the integrated system against specified requirements, culminating in acceptance testing to ensure that the system meets the end user needs and operational criteria. The V-model distinguishes itself by mirroring each development activity with a corresponding testing activity, allowing for parallel testing and early error detection. This methodical approach ensures a disciplined and rigorous process that facilitates early identification and correction of errors, underscoring its preference for projects that require high levels of reliability and safety. The V model, along with certification standards such as DO-178C, plays a crucial role in the assurance processes for the development of aviation software. This model provides a systematic framework that delineates the development and testing phases in a linear sequence, progressing from requirement specifications to coding and subsequently through various testing stages in reverse order. In contrast, DO-178C, also recognized as the Software Considerations in Airborne Systems and Equipment Certification, delineates the criteria for avionics software development to ensure compliance

with essential safety and performance benchmarks.

However, when it comes to incorporating machine learning (ML) technologies, these traditional paradigms face notable challenges. The inherent determinism of the V model and the rigorous demands of DO-178C clash with the stochastic and data-driven nature of ML models. This discordance introduces a pronounced gap in the assurance framework for aviation systems that utilize machine learning, as the pre-existing methodologies fall short of comprehensively addressing the distinct verification and validation requirements of ML-driven applications. This gap underscores the imperative to revise current standards and formulate novel methods to facilitate the safe and effective integration of machine learning technologies into aviation systems.

In examining these classical frameworks, it becomes evident that traditional assurance processes in aviation are not adequately prepared to face the challenges posed by machine learning technologies. The divergence primarily stems from the traditional reliance on deterministic outcomes, whereas machine learning introduces a layer of probabilistic outcomes and data dependence that current standards do not sufficiently cater to. This discrepancy accentuates the need for innovative assurance methods for ML applications within aviation, pinpointing the substantial gap that my research intends to bridge.

1.3 Motivation for this Research Work

Software quality assurance is a well-established practice for traditional (control-driven) software systems, with numerous methods extensively used in the industry. However, as MLSs has a fundamentally different programming paradigm and decision logic representation, existing quality assurance approaches can hardly be used for them in an as-is fashion. ML models learn from a given set of scenarios. What should the requirements be for the definition of these scenarios? The absence of such requirements casts serious doubts on the training and assurance of ML components embedded safety critical domains. Most of the existing MLSs training procedures only reveal and measure training loss and validation loss, in terms of prediction accuracy. Root cause analysis becomes especially challenging when the training method changes, which might occur due to upgrades in the ML libraries or changes in the training data set or in the system platform itself.

In the last few years, ML has become an indispensable part of systems that involve humans or operate in shared environments with humans, resulting in growing attention to the aspects of the safety of those systems. Edge nodes and Internet of Things (IoT) systems present new

fronts where ML techniques and technologies might be leveraged. The deployment of conventional ML however requires enormous amounts of power and resources.
MLSs behavior is heavily influenced by factors such as the available training data sets, hyperparameters, model architecture, algorithm and optimization technique (to find the optimal set of parameters that result in the best performance of the model on the given task, such as improving accuracy or reducing error) [12]. The source code of MLSs is typically concise and less error prone since it consists of calls to the application programming interface (API) performing highly sophisticated high-level functions, and the decision-making policy (i.e. correlations) is inferred from the training data. However, the behavior of MLSs encoded in the learned weights of the model, is difficult for humans to interpret and debug. Classical testing techniques are not easily applicable to MLSs artifacts, except for the source code. Therefore, testing frameworks for MLSs require additional approaches to address these challenges.
The motivation for assuring Machine Learning (ML) systems enabled by TinyML for Avionics Systems is grounded in the need for ultra-reliable, efficient, and secure computational solutions in the highly demanding environment of aviation. TinyML, with its focus on deploying lightweight ML models on low-power, compact hardware, offers a promising avenue for enhancing various aspects of avionic systems, from predictive maintenance to in-flight monitoring and control [30]. The assurance of these systems is crucial due to the critical nature of aviation operations, where the margin of error is minimal, and the consequences of failure can be catastrophic. It involves ensuring that these TinyML implementations not only meet the stringent safety and reliability standards of the aviation industry but also maintain consistent performance under the unique challenges of the avionic environment, such as limited computational resources, varying temperatures, and potential electromagnetic interference [31]. The process also tackles security issues related to ML's incorporation into critical aviation systems, an area not fully explored in existing reviews. This gap highlights the need for a robust assurance framework for ML systems in aviation, a challenge this dissertation aims to address, marking a significant advancement in the field.

1.4 Research Gaps

The state of the art shows that software quality assurance is a well-established practice for traditional (control-driven) software systems, with numerous methods extensively used in the industry. However, as MLSs has a fundamentally different programming paradigm and decision-logic representation, existing quality assurance approaches can hardly be used for them in an

as-is fashion. ML models learn from a given set of scenarios. The challenges that may be posed in this way are diverse, ranging from sensing, estimation, and control to machine learning and decision making under uncertainty proposed by Shapiro, Dentcheva, and Ruszczynski [32]. A notable research gap exists to ensure the robustness and reliability of Machine Learning models within these systems. Avionic systems, integral to aircraft operation and safety, require absolute precision and fault tolerance. The integration of Machine Learning (ML) models into aviation systems presents significant opportunities for improving efficiency, enabling predictive maintenance, and facilitating real-time decision-making. The potential of ML models in this domain is vast, offering a transformative impact on how aviation systems operate and respond to varying conditions [33]. However, the dynamic and complex nature of the aviation environment introduces specific challenges for these ML models. Factors such as variable flight conditions, the diversity of sensor data, and the paramount requirement for continuous, fail-safe operation pose significant hurdles. These models must be adept at handling these complexities while maintaining high accuracy and reliability [34]. Currently, the aviation industry faces a gap in the development of comprehensive frameworks and methodologies that are specifically designed to evaluate and assure the performance of ML models in such demanding avionic conditions. This gap underscores the need for dedicated research and development efforts to establish robust evaluation and assurance protocols that can effectively address the unique requirements of avionic This leads us to the following open issue.

- How to provide the Assurance of Machine Learning based systems for Avionics Systems?

Firstly, the literature review showed that ML models are mostly tested relying on accuracy metrics in large test sets that attempt to represent a meaningful subset of the targeted operational domain. This is insufficient when ML is deployed within a real-world system, as the examples gathered in the field may differ substantially from those of the test set.
Second, practitioners in the field of safety-critical systems, particularly in the aviation industry, face the significant challenge of the absence of comprehensive guidelines for designing architectures, especially for integrating AI and ML technologies. While these technologies hold immense potential, their integration into systems where safety is paramount is a complex endeavor. The aviation sector, along with other industries aiming to implement AI/ML for autonomous operations, faces this challenge acutely [35]. A key obstacle in this pursuit is addressing a range of safety-related concerns, chiefly the assurance of safe autonomous functioning of AI/ML-enabled systems. This difficulty is intensified by the inherent traits of AI/ML technologies,

which can vary from being non-deterministic and unpredictable to complex and challenging to interpret. Such characteristics make the task of ensuring reliability and safety in AI/ML-integrated systems particularly daunting, necessitating a focused and specialized approach to system design and validation [36].

1.5 Research Questions

This research study was focused on incorporating TinyML-aware solutions for Machine Learning Systems (MLSs) in safety-critical environments, aiming to achieve an optimal balance between the size, accuracy, and performance of the ML model, and the constrained capabilities and stringent energy efficiency requirements of the MLSs in executing these models. The core objective was to ensure that these implementations provided adequate assurance, a critical factor in safety-critical systems where reliability and precision are paramount. This approach addresses the need to adapt ML technologies to the unique demands of safety-critical systems, providing solutions that are both efficient and reliable within the limited operational scope of such systems.Specifically, we have established the following three research questions.

- RQ-1: What specific standards and conditions must be established for crafting training scenarios for Machine Learning models to guarantee thorough training and consistent, reliable performance across a wide range of practical applications, including control-loop systems in avionics?

- RQ-2 : What methodologies and measures can be implemented during the development phase of Machine Learning models in avionics to effectively minimize the risk of performance insufficiencies and ensure robust functionality when deployed in open-world environments?

- RQ-3: How does embedding Machine Learning (ML) in avionic systems contribute to enhancing model assurance, particularly by boosting reliability, safety, and instant decision-making, despite the aviation industry's limited computational capacity and strict regulatory requirements?

Due to the rate of development of MLSs and the potential benefit from deploying such systems, these questions require urgent answers in the avionics context.

1.6 Research Contributions

This thesis explores these questions, for example identifying safety challenges posed by ML from a general and a regulatory perspective. The thesis then presents technical solutions which go some way towards answering those questions. More specifically, the contribution of this thesis is three-fold:

1. Defining training scenarios for Machine Learning models, operational domain design is key, focusing on identifying and incorporating a broad spectrum of real-world conditions and variables. This approach ensures the models are trained on diverse, representative datasets, covering the full range of operational scenarios they are likely to encounter in avionics systems. As a result, the models exhibit comprehensive learning and robust performance across various applications.This addresses question 1.

2. The novel method FAST-TinyML-DNN classifier (FTD) was propsed during the process of development of ML models, focusing on Out-Of-Distribution (OOD) detection. FTD, tailored for TinyML's specific computational and storage limitations, aims to improve the model's capability to accurately identify and manage unexpected data.This addresses question 2.

3. The novel framework MELOD, which stands for Multi-Layer Early Exit for OOD Detection with LOF in DNNs. This approach, optimized for TinyML, integrates a multi-layer early exit technique in Deep Neural Networks (DNNs) for effective out-of-distribution detection. It involves embedding LOF-based outlier detection within DNN hidden layers, enhancing the network's ability to efficiently identify and process anomalous data. This framework directly tackles the issues raised in the second and third questions by leveraging TinyML's capabilities.

In acknowledging the journey of this research, it's important to note that while there have been submissions made, they have not yet met with success. However, the materials developed and refined through these submission attempts have become integral parts of this thesis and are poised for successful submission in the near future. This progression reflects the iterative nature of academic research and the ongoing refinement of ideas and methodologies.

The key elements derived from these submission efforts are embedded throughout the thesis document. Specifically, the refined research methodologies, developed after feedback from initial submissions, can be found in Chapter 3, titled " Objectives, Methodology, and Results. Additionally, the Software Testing Methods and Strategies for Machine Learning-Based Systems: a Systematic Literature Review, is thoroughly discussed in Chapter 2, "Background".

1.7 Thesis Structure

The remaining of this thesis is organized in the following chapters

- Chapter 2 (Background and Related Work): It provides background information about the related concepts used in this work to answer our research questions, as mentioned in the previous section. The material forming Chapter2 of this thesis is reflective of the extensive work conducted in preparation for journal submissions. This includes substantial research and development that contributed to the paper titled:

    1. "Software Testing Methods and Strategies for Machine Learning-Based Systems: A Systematic Literature Review," co-authored by Iqbal, Z., and Vardanega, T., which is targeted for submission in 2024 to the Software Testing, Verification and Reliability (STVR) Journal. The insights and methodologies developed through this process are thoroughly integrated into the aforementioned section of the thesis, showcasing a deep understanding and application of advanced software testing strategies for ML-based systems.

- Chapter 3 (Objectives, Methodology, and Results): In this chapter, we first state three research objectives which were formulated by keeping in view our research questions and the background knowledge of the related concepts, as mentioned in Chapter 2. Next, we provide details of the experimental studies and methods used. Subsequently, we discuss our methodology and the obtained results for all research objectives.The content within Chapter 3 of this thesis encapsulates the extensive research and development efforts undertaken in preparation for various academic and professional platforms, including workshops, conferences, and journal publications as following:

    1. This initial conference paper focuses on the foundational research that underpins "Out-of-Distribution Detection in Machine Learning Based Control Systems enabled by TinyML," authored by Iqbal, Z., and Vardanega, T., which has been submitted for review at the 33rd International Conference on Artificial Neural Networks (ICANN). The methodologies, insights and findings detailed in this section of the thesis are integral to the paper and are the result of meticulous planning and strategic execution.
    2. The second paper will include significant contributions to the "Role of Operational Design Domain in Out-of-Distribution Detection for Avionics enabled by TinyML," authored by Iqbal, Z., and Vardanega, T., scheduled for submission in 2024 IIEEE Symposium on Visual Languages and Human-Centric Computing. This paper is a key component of my research portfolio, and the insights and results within it are directly reflected in the specified section of the thesis

3. The third paper titled "Introducing AI to IMA Technology – System Perspective," authored by Iqbal, Z., Lehmann, M., Lüttig, B., Ayyildiz, R., Bobrzik, T., Vardanega, T., and Daw, Z., utilizes innovative techniques in TinyML for optimizing cabin pressure control within Integrated Modular Avionics (IMA) systems. This work was submitted to the 43rd Digital Avionics Systems Conference (DASC) and has been accepted for a lecture presentation at the conference, showcasing its significance and contribution to the field of avionics and AI integration.

The preparation for these submissions has been meticulously planned, with clear targets and timelines for each submission to relevant conferences and journals.

- Chapter 4 (Conclusions and Outlook): This final chapter contains our conclusions and a summary of the contributions made. Then, we present a detailed discussion on our methodology and obtained results, including their limitations and recommendations for potential improvements. Finally, we discuss the future work directions of this research work.

2

Background

In this chapter, we present a background overview of the key topics explored in this doctoral research work. This chapter sets the background to our research project. The chapter is divided into two sections, each of which introduces one of the two principal contextual elements of the problem space: the ambit of safety-critical systems and of avionics systems in particular, with their distinguishing characteristics; and the essentials of the Machine Learning domain, seen from the perspective of embedding ML models in the command-and-control part of avionics systems. Section 2.1 in particular focuses on the concept of Operational Design Domain (ODD), which is bound to play an increasingly important role in the verification-based provision of assurance in safety-critical systems, enhancing the implementor's ability to conform with the applicable certification standards with greater precision and agility. Section 2.2 delves into the core of Machine Learning (ML) and its burgeoning role in revolutionizing the way avionics industry is going to develop its new-generation systems. That section discusses the main obstacles inherent in the integration of ML into aviation systems, with special focus on the challenge of detecting out-of-distribution (OOD) situations referring to scenarios where an ML system encounters data that significantly deviate from the data it was trained on. In the context of aviation, this is a critical challenge because the safety and reliability of flight systems depend heavily on their ability to handle unexpected or novel situations accurately. This discussion naturally leads to observe that crucial role that TinyML may have in helping ML enter avionics, striking a good balance between the push of innovation and the pull of preservation of conformance with stringent and conservative safety regulations in the application domain.

2.1 Safety-critical systems

Systems deemed as safety-critical play a crucial role in various sectors, including transportation (avionics, railways, and automotive), space, telecommunication, civil and military infrastructure (such as nuclear and power plants), and medical and control devices. These systems are essential because their malfunctioning could result in severe harm, including loss of life, or substantial damage to property and the environment.To mitigate these risks, safety-critical systems are developed under the guidance of specific certification standards. These standards impose on developers a comprehensive set of obligations that cover all aspects of the development process.

Software within these systems is increasingly becoming a focal point. As the demand for systems to perform a wider range of tasks increases, the size and complexity of the software escalate accordingly to meet these demands [37]. While safety-critical systems frequently embed specialized hardware parts, which can take the largest fraction of the development cost, their software accounts for an increasing proportion of the delivered functionality and has a disproportionately large impact on overall safety. This is evidenced by numerous incidents attributed to software failures, highlighting its reliability as a critical factor in the dependability of the system as shown in Figure 2.1. Consequently, the costs associated with software development and evaluation are among the most significant ones, and on the verge to dominate the entire system lifecycle [38]. This scenario has spurred researchers and practitioners to focus intensely on improving these aspects. In the field of avionics, systems classified as safety-critical are those whose failure could result in catastrophic outcomes such as loss of life, significant property damage, or severe environmental impact [39]. This high-stakes environment requires unparalleled reliability in software. In avionics, even minor software glitches can precipitate complete system failures, potentially leading to aviation disasters. This critical nature of avionics systems highlights the vital importance of rigorous software quality management. Ensuring that avionic software is not only functional but also adheres to the highest standards of safety and reliability is not just a technical requirement but a moral imperative.

2.1.1 Avionics Systems

Avionics systems, integral to the aerospace industry, are quintessential examples of safety-critical systems due to their direct impact on the safety of flight operations. These complex systems encompass navigation, communication, and flight control functions, and any failure within

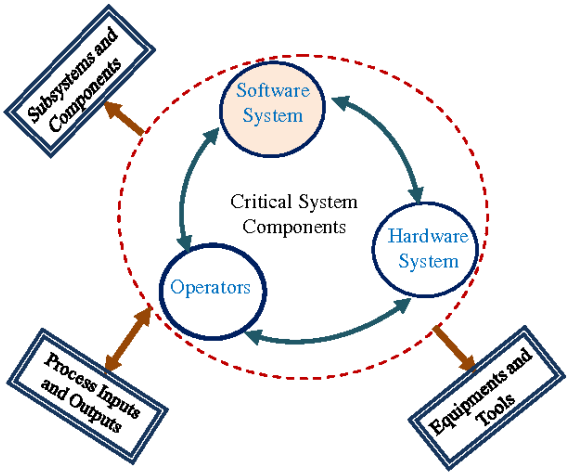

Figure 2.1: A diagram illustrating a safety-critical system, featuring the operator, software system, and hardware system, along with detailed annotations of process inputs and outputs, subsystems, components, equipment, and tools essential for ensuring operational integrity and safety [5].

them can have dire consequences, including risks to human life and potential damage to aircraft or the environment. The intricate nature of avionics demands stringent adherence to safety protocols and rigorous testing to ensure reliability and fault tolerance[5]. Given the high stakes, avionics systems are designed with multiple redundancies and robust fail-safe mechanisms. The development of these systems follows rigorous standards and certifications, reflecting the paramount importance of safety in all aspects of their design and operation. This criticality also drives continuous innovation in avionics, with the aim of improving safety features while accommodating the evolving demands of modern aviation.

The role of assurance in avionics, as a safety-critical system, is both integral and multifaceted, focusing primarily on ensuring system reliability and safety.

In this context, "system reliability" refers to the ability of the avionics systems to perform their intended functions consistently and accurately over time. This means the systems are expected to operate without failure, delivering the correct outputs and responses under various conditions. Reliability is crucial in avionics because consistent performance is key to maintaining operational effectiveness and preventing accidents or malfunctions in flight.

"System safety," on the other hand, relates to the avionics systems being free from conditions that could lead to significant hazards or accidents. This involves designing systems in a way that either eliminates or significantly reduces the risk of such occurrences. Safety in avionics is paramount due to the high stakes involved - any failure can have severe, potentially catastrophic, consequences. In this context, assurance refers to a set of systematic, documented

processes and procedures designed to ascertain that avionics systems meet stringent safety requirements and function as intended under all circumstances. According to Leveson in [40], assurance in avionics involves rigorous safety analyzes, thorough tests, and validation processes to identify and mitigate potential system failures. Additionally, in [41] Stolzer, Halford and Goglia emphasize the importance of adherence to established international standards, such as those set by the Federal Aviation Administration (FAA) and the European Union Aviation Safety Agency (EASA), which dictate specific safety and certification guidelines for avionics [42]-[43]. Furthermore, as Rushby highlighted in [44], continuous monitoring, regular updates, and maintenance are vital to ensure long-term reliability and safety of avionics systems, reflecting the dynamic nature of technological and operational environments in aviation. The assurance process in avionics involves comprehensive testing, validation, and certification protocols to ensure that every component functions flawlessly under a wide range of conditions. Furthermore, as reported by Endsley and Jones in [45], the integration of advanced technologies in avionics, like automated systems and machine learning , has raised the bar for assurance, necessitating even more robust and sophisticated validation methods to address the complexities of these systems. The importance placed on assurance highlights the crucial role of avionics in ensuring the overall safety and effectiveness of air travel.

The avionics system lifecycle is a sophisticated and rigorously structured process that is integral to ensuring the safety and efficiency of aircraft systems. This lifecycle is traditionally divided into several key phases: requirements analysis, system design, implementation, testing,deployment and maintenance. Within this framework, the concept of the Operational Design Domain (ODD) plays a critical role, as it defines the specific operational conditions under which avionic systems are expected to function. According to Ferrell and Leveson in [46], the ODD encompasses various environmental, geographic, and operational scenarios that an aircraft might encounter, and it is essential for guiding the development process. During the requirements analysis phase, engineers and designers determine the functional and performance requirements of the avionic systems within the context of the ODD. As Newman highlighted in [47], the design and implementation phases must then integrate these requirements to develop systems that are robust and adaptable to the defined conditions. The testing phase, crucial for validating the functionality and safety of the systems within the ODD parameters, involves rigorous simulations and real-world trials. Finally, the maintenance phase ensures that the systems remain effective and safe over time, adapting to any changes in operational requirements or environments. This lifecycle approach, with a focus on ODD, ensures that avionic systems are not only technologically advanced, but also reliable and safe for their intended use.

2.1.2 Avionics control systems

Avionics control systems represent the heart of modern aircraft, encompassing a wide array of electronic systems that perform individual functions ranging from navigation and communication to the monitoring and management of multiple flight control systems. These systems are critical for ensuring the safety, efficiency, and reliability of aircraft operations, both in commercial and military aviation. The integration of advanced technologies into avionics control systems has been a continuous pursuit within the aerospace industry, aiming to improve performance and safety standards. The evolution of these systems has been marked by significant milestones, notably the transition from analog to digital systems and the increasing incorporation of software-driven functionalities, which have dramatically enhanced the capabilities of avionics systems [48].

Machine Learning technologies have emerged as a transformative force in the development of next-generation avionics control systems. By leveraging ML algorithms, avionics systems can now process and analyze vast amounts of data in real-time, enabling more precise control, predictive maintenance, and adaptive response mechanisms. These advancements facilitate improved decision-making processes, optimizing flight operations and enhancing overall aircraft performance. However, the integration of ML into safety-critical systems such as avionics presents unique challenges, particularly regarding the assurance of system reliability and safety under all operational conditions [49].

The concept of Tiny Machine Learning (TinyML) has gained attention as a promising solution to overcome some of the challenges associated with deploying ML in resource-constrained environments like avionics control systems. TinyML refers to the optimization and implementation of lightweight ML models that can run on low-power microcontrollers, offering the potential to embed intelligent functionalities directly into various avionics components without significantly increasing power consumption or system complexity [50]. This approach not only supports the real-time processing requirements of avionics systems but also opens up new possibilities for enhancing system capabilities, such as improved anomaly detection, fault diagnosis, and autonomous decision-making.

Despite the promising advantages of TinyML in avionics, ensuring the reliability and safety of ML-based systems remains a paramount concern. The development and validation of these systems require rigorous testing and verification processes to meet stringent aviation standards. The adoption of a structured development model, such as the W-shaped model discussed earlier, can provide a comprehensive framework for integrating TinyML into avionics control sys-

tems. This model emphasizes a thorough analysis of system requirements, iterative design and testing phases, and continuous validation to ensure that the integrated ML models are robust, reliable, and capable of performing as intended under all operational scenarios [51].

In conclusion, the integration of ML, particularly through TinyML, into avionics control systems represents a significant advancement in aerospace technology. It holds the promise of making aircraft more intelligent, efficient, and safe. However, the realization of this potential requires overcoming significant technical and regulatory challenges. Ongoing research and development efforts, guided by robust development frameworks, are essential to address these challenges, paving the way for the successful implementation of ML in avionics systems.

2.1.3 Role of Operational Design Domain in Avionics safety

The concept of Operational Design Domain (ODD) is integral to understanding the operational parameters within which a system is designed to function, originally established in the automotive industry. The term "ODD" refers to specific conditions under which a system, like an autonomous vehicle, operates effectively. This concept has been thoroughly explored in the automotive sector, with a notable taxonomy established in 2020, as referenced in [52]. Studies, as cited in [53]-[54]-[55], have emphasized the importance of ODD for safety in the automotive industry.

In aviation, however, the application of ODD is still emerging. The European Union Aviation Safety Agency (EASA) documented the first application of ODD in aviation, offering a detailed definition in their publication[56]. Here, ODD for Machine Learning Systems (MLSs) in aviation is characterized as the range of operating conditions, including environmental factors, geographical settings, and time-of-day restrictions, under which the system is intended to function optimally. The ODD for MLSs in aviation is not just a static set of parameters; it considers the interrelationships between different operating conditions. For instance, the acceptable range for certain parameters may vary depending on other concurrent conditions, allowing MLSs to operate more effectively in a dynamic, real-world environment. To provide a deeper understanding of Operational Design Domain (ODD) parameters in aviation, let's explore some specific examples. These parameters define the conditions and situations under which an aviation system [57], particularly those using MLSs, is designed to operate effectively and safely.

1. **Weather Conditions**: ODD parameters may include a range of weather conditions under which the system can operate reliably. For instance, an MLSs might be designed to

function optimally in conditions ranging from clear skies to moderate rain. However, severe weather phenomena like thunderstorms or heavy snowfall might fall outside its ODD.

2. **Geographical Areas**: This could specify certain types of geographical settings where the system is expected to operate. For example, an MLSs in a commercial aircraft might be optimized for operation over continental landmasses, but not for polar routes or certain mountainous terrains.

3. **Air Traffic Density**: ODD parameters can also define the level of air traffic the system is designed to handle. For a system used in air traffic control, its ODD might encompass operating efficiently in medium to high traffic volumes but may not be designed for extremely congested airspace scenarios.

4. **Altitude Ranges**: This involves the range of altitudes at which the system can function effectively. A system might be designed for cruising altitudes typical of commercial flights but not for the lower altitudes used during takeoff and landing phases.

5. **Time-of-Day Restrictions**: Some systems might be optimized for operation during daylight hours but not equipped to handle the different challenges presented by nighttime operations.

6. **Communication System Reliability**: The ODD might include the expected reliability and availability of communication systems. For instance, a system may be designed to operate effectively with certain levels of satellite communication availability but not in scenarios where communication signals are weak or interrupted.

7. **Emergency Situations**: The system's ODD might include its response to standard emergency procedures but not be designed for highly unusual or catastrophic emergency scenarios.

By defining these parameters, avionics engineers can ensure that the MLSs is equipped to handle expected operational scenarios while also being aware of its limitations. Understanding these limits is crucial for maintaining safety, as it informs the development of contingency protocols for scenarios that fall outside the ODD. This nuanced approach to ODD in aviation is critical, especially when integrating ML components. ML-based system in aviation must reliably operate across diverse scenarios, as emphasized by Ferrell and Leveson in [46]. By defining the ODD, engineers can tailor ML algorithms to meet the stringent safety requirements prevalent in the aerospace industry, a point highlighted by Newman in [47].

However, the principles and guidelines established for ODD in the automotive sector cannot be directly applied to aviation due to several key factors. These include different regulatory

approaches, the necessity for aviation standards to align with existing regulations, higher levels of assurance requirements in aviation, and the need for consistency with established aviation-specific engineering practices, such as safety assessment and system development processes cited in [58] and [59].

To illustrate, let us consider an example of ODD in the context of an autonomous flight system. The ODD may define operational limits such as specific weather conditions (e.g., clear skies vs. thunderstorms), types of airspace (e.g., urban vs. rural), and altitude ranges. Within these defined limits, the ML system is expected to perform optimally. However, if the system encounters conditions outside of these defined parameters, like unexpected severe weather or emergency airspace restrictions, it would be considered operating outside of its ODD. This necessitates robust systems for detecting and responding to such out-of-ODD scenarios to maintain safety and reliability.

2.1.4 Machine Learning & the Operational Design Domain: Enhancing System Assurance and Safety

Operational Design Domain delineates the specific operational conditions under which an ML system is designed to function reliably and safely. This concept, originating from the autonomous vehicle industry, has rapidly become a cornerstone in the broader field of ML safety and reliability. As Koopman and Wagner noted in [60], ODD encompasses a range of environmental, geographical, temporal, and other operational constraints that define the safe operational parameters of an ML system. By establishing these boundaries, ODD serves as a critical tool for system developers and engineers to ascertain the limits within which ML-based system can make decisions, react to input and operate effectively without human intervention.

The significance of ODD in assuring ML-based system stems from its role in bridging the gap between theoretical performance and real-world applicability. Traditional ML models are often trained and tested in controlled or idealized environments, which may not fully represent the complexity and unpredictability of real-world scenarios. ODD provides a structured framework for testing and validating ML-based system under more realistic conditions, ensuring that they can handle the complexities and uncertainties inherent in their intended operational environments. This is particularly crucial in high-stakes domains such as autonomous driving, aviation, and healthcare, where the cost of failure can be catastrophic. As highlighted by Shalev-Shwartz et al. in [61], the integration of ODD in the development process allows a more targeted approach to model training and validation, focusing on the specific conditions

and challenges the system is expected to encounter. This targeted approach not only enhances the robustness and reliability of ML-based system, but also aids in building trust among users and regulators, a key aspect in the widespread adoption of autonomous technologies.

Furthermore, the role of ODD extends beyond the development phase, contributing significantly to the ongoing assurance and adaptability of ML-based system. As systems are deployed in dynamic real-world environments, they encounter novel situations and variables that were not present or considered during the training phase. Therefore, the concept of ODD plays a vital role in the continuous monitoring and adaptation of these systems. In this context, the work of Gauerhof et al. in [62] becomes relevant, emphasizing the need for dynamic ODDs that can evolve based on the accumulation of operational data and experiences. This adaptive approach ensures that ML-based system remain within their safety and performance thresholds even as they encounter new scenarios, a concept aligned with the broader goals of AI safety and lifelong learning in ML. In addition, ODD is instrumental in regulatory compliance and certification processes, providing a clear and tangible framework against which ML-based system can be evaluated. Regulatory bodies and certification agencies require concrete evidence that ML-based system can operate safely within their intended domains. By defining ODD, developers can offer this evidence, demonstrating that their systems have been rigorously tested and validated within these domains. This aspect of ODD is particularly highlighted in [63] the work of Varshney and Alemzadeh, who discuss the challenges and methodologies for certifying AI systems in healthcare, a sector where safety and reliability are paramount. In essence, ODD serves as a critical link between the theoretical capabilities of ML-based system and their practical and safe deployment in real-world scenarios.

Analyzing the impact of input data that fall outside the ODD is crucial for the effective functioning of ML-based system, particularly in avionics and autonomous flight operations. To mitigate such risks, continuous monitoring of input data is essential to determine whether it falls within the defined ODD. If the data is detected to be outside of the ODD, the system must have robust protocols to correct the course of action or seek alternative decision-making pathways, such as human intervention.

2.2 Machine Learning

In the realm of Machine Learning (ML), data is the cornerstone upon which algorithms operate and derive knowledge. Unlike rule-based programming, ML algorithms depend on the analysis of data to uncover patterns and make predictions [64]. The effectiveness of these al-

gorithms is deeply influenced by the quality and quantity of the data they are fed. Broadly, datasets in ML can be categorized into two types:

1. **Unlabelled Dataset**: This dataset type is defined as $X = \{x^{(n)} \in \mathbb{R}^d\}_{n=1}^N$. It consists of a feature set X containing N samples, where each sample is a d-dimensional vector. These vectors are also known as feature vectors or samples. In this context, each dimension of a vector is referred to as an attribute, feature, variable, or element. Unlabelled datasets do not have associated output labels or targets.

2. **Labelled Dataset**: This dataset type is represented as $X = \{x^{(n)} \in \mathbb{R}^d\}_{n=1}^N$, $Y = \{y^{(n)} \in \mathbb{R}\}_{n=1}^N$. It not only includes the feature set X but also a label set Y, which records the corresponding label for each feature vector. An alternative representation of a labelled dataset can be $\{(x^{(n)}, y^{(n)}) \in \mathbb{R}^d \times \mathbb{R}\}_{n=1}^N$, where each pair $(x^{(n)}, y^{(n)})$ is known as a data pair.

In the context of ML, the goal is to find a function $g(x)$ that closely approximates an unknown target function $y = f(x)$, where for any $x \in X$, the function f accurately outputs the corresponding y in the label space. The dataset utilized to learn this function $g(x)$ is referred to as the training set or training data. In contrast, the test set or test data, which is distinct from the training set, is used to evaluate the performance of $g(x)$. The distinction between these two sets is crucial, as it allows for the assessment of the model's ability to generalize to new, unseen data, a fundamental aspect of ML efficacy as shown in the Figure 2.2.

2.2.1 Categories of Machine Learning

ML methodologies are broadly categorized into five key areas: supervised, unsupervised, semi-supervised, self-supervised, and reinforcement learning. Each category serves distinct purposes and is suited to specific kinds of data and learning tasks. In this thesis, we focus primarily on supervised and self-supervised learning, as these methodologies align closely with the nature of our problem and the tasks at hand.

- **Supervised Learning**: Supervised learning [65] is characterized by its use of labeled datasets. The objective is to discern the relationships between the feature set (input data) and the label set (output data), extracting knowledge and properties from this labeled dataset. There are two main types of supervised learning tasks:

    1. Classification: If each feature vector x corresponds to a discrete label y in a label set $L = \{l_1, l_2, \ldots, l_c\}$, where c can range from 2 to several hundred, the task is termed classification. Here, the goal is to categorize each input data point into one of the predefined classes.

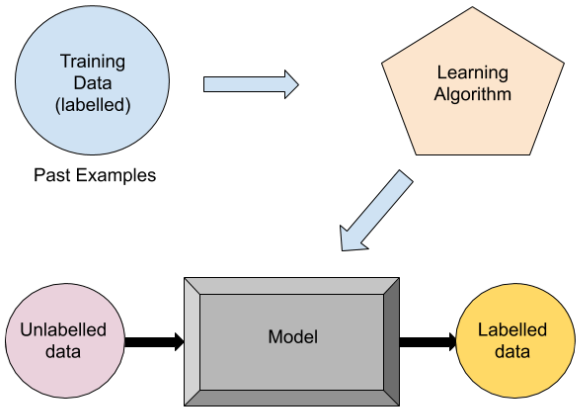

Figure 2.2: The figure illustrates the process of training a machine learning model using both labeled and unlabeled data, showcasing the steps of data classification and model training.

2. Regression: In cases where each feature vector x corresponds to a real value $y \in \mathbb{R}$, the task is known as a regression problem. The focus is on predicting a continuous quantity based on the input features.

Knowledge gained from supervised learning is typically used for prediction and recognition tasks.

- **Self-Supervised Learning**: Self-supervised [66] learning, a relatively new paradigm in ML, is a subset of supervised learning but with a unique twist. In self-supervised learning, the data labels are not provided explicitly. Instead, the algorithm generates its own labels from the input data. This is typically done by designing a pretext task, where the model predicts part of the data given other parts. For example, a common pretext task in image processing is to remove a portion of the image and train the model to predict the missing part. The key advantage of self-supervised learning is that it leverages unlabeled data, which is more abundantly available, and learns robust feature representations that can be useful for downstream tasks [67].

- **Unsupervised Learning**: Although not the focus of this thesis, unsupervised learning warrants a brief mention [68]. It deals with unlabeled datasets, exploring and drawing inferences from this data to identify hidden patterns or structures. Common applications include clustering, probability density estimation, feature association, and dimensionality reduction. The insights gained from unsupervised learning can sometimes be beneficial in enhancing supervised learning models.

- **Semi-Supervised Learning**: represents a middle ground in ML methodologies, bridging the gap between supervised and unsupervised learning [69]. It is particularly relevant in scenarios where obtaining a large set of labeled data is challenging or costly, but where ample unlabeled data is available. Semi-Supervised Learning leverages both a small amount of labeled data and a larger volume of unlabeled data for training. This approach is based on the assumption that the patterns or the distributions discovered in the unlabeled data can significantly contribute to the learning process, enhancing the model trained with the limited labeled data.In semi-supervised learning, the algorithm initially learns from the labeled data, just as it would in a supervised learning context. However, it then extends this learning to the unlabeled data [70]. Semi-Supervised Learning offers a pragmatic solution in ML, particularly in scenarios with limited labeled data. By combining the strengths of both supervised and unsupervised learning, it provides a versatile framework for efficiently developing robust models.

- **Reinforcement Learning (RL)**: stands as a unique and powerful branch of Machine Learning, centered on the interaction between an agent and its environment [71]. The essence of RL lies in the agent's ability to make decisions and learn from the consequences of its actions, rather than from explicit instruction. In this framework, the agent explores its environment, performs actions, and receives rewards or penalties in response. These rewards serve as critical feedback, guiding the agent to refine its policy - a set of rules or strategies dictating its actions. Over time, through a process of trial and error, the agent learns to maximize cumulative rewards, effectively shaping its behavior towards achieving optimal or near-optimal outcomes. RL's strength is its applicability to a wide range of complex problems, from game playing and robotics to autonomous vehicles, where learning through interaction is key. This approach fosters not just the replication of known strategies but the discovery of novel solutions to challenges, making RL a frontier field in the development of intelligent, adaptive systems.

2.2.2 Key Elements of Machine Learning

In the broad area of machine learning (ML), irrespective of the particular technique employed, whether it is supervised, unsupervised, or reinforcement learning, three essential elements consistently play a vital role in the construction and operation of ML-based system. These components, as identified in [72], are Representation, Evaluation, and Optimization, each playing a pivotal role in the learning process.

- **Representation**

  The representation aspect of ML deals with how knowledge is encoded within the system. This step is crucial as it defines the hypothesis space H, which encompasses all possible hypotheses h that the learning algorithm can formulate. These hypotheses are essentially different mapping functions or distributions the algorithm considers in its learning process. The goal of ML in this context is to identify the most accurate hypothesis h, known as the final hypothesis, that approximates the target function best. Various types of representations include neural networks, decision trees, probabilistic graphical models, and support vector machines. The choice of representation significantly influences the types of patterns and relationships the algorithm can learn and detect.

- **Evaluation**

  Evaluation in ML refers to the method of assessing and scoring candidate hypotheses. This is achieved through an evaluation function, also known as an objective function, utility function, loss function, scoring function, or fitness function, depending on the context. This function is vital to differentiate between hypotheses and to guide the learning process towards more accurate predictions. Common examples of evaluation functions include mean squared error in regression tasks or likelihood functions in probabilistic models. The choice of evaluation function has a substantial impact on the preferences and priorities of the learning algorithm within the hypothesis space.

- **Optimization**

  The third component, Optimization, involves the search strategy employed to find the highest-scoring hypothesis in the hypothesis space. It is the process of tweaking and adjusting the parameters of the ML model to minimize or maximize the evaluation function, thereby enhancing the model's performance. The optimization technique chosen is crucial for the efficiency and effectiveness of the learning process. Popular optimization methods include stochastic gradient descent, commonly used in neural networks, and greedy search, often employed in decision tree algorithms. It's important to note that once a model is trained, the specific details of the optimization process may not be retrievable, but the impact of this process is reflected in the model's performance.

In summary, the synergy between Representation, Evaluation, and Optimization defines the core of any ML system. These components collectively determine how an ML algorithm processes data, learns from it, and makes predictions or decisions, thus shaping the capabilities and effectiveness of the ML solution.

2.2.3 Performance Metrics for ML Models

Evaluating the performance of ML models is a critical step in determining their effectiveness and suitability for a given problem. Various metrics are used to assess different aspects of a model's performance, with accuracy, precision, recall, F1 score, and the AUC-ROC (Area Under the Receiver Operating Characteristic Curve) measure being among the most commonly used, especially in the context of deep learning.

- **Accuracy**

  Accuracy is a fundamental metric that measures the overall correctness of the model. It is calculated as the ratio of the number of correct predictions to the total number of predictions made by the model. Formally, it is represented as:

  $$\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN}$$

  where:

  - TP (True Positives) are correctly predicted positive observations.
  - TN (True Negatives) are correctly predicted negative observations.
  - FP (False Positives) are negative observations incorrectly predicted as positive.
  - FN (False Negatives) are positive observations incorrectly predicted as negative.

- **Specificity**

  Specificity, also known as the true negative rate, is given by:

  $$\text{Specificity} = \frac{TN}{TN + FP}$$

  It measures the proportion of actual negatives that are correctly identified.

- **Precision**

  Precision indicates the accuracy of positive predictions. It is defined as:

  $$\text{Precision} = \frac{TP}{TP + FP}$$

  It represents the proportion of positive identifications that were actually correct.

- **Recall**

  Recall, also known as sensitivity or the true positive rate, is defined as:

$$\text{Recall} = \frac{TP}{TP + FN}$$

  Measures the proportion of actual positives that are correctly identified.

- **F1 Score**

  The F1 score is a harmonic mean of precision and recall, providing a balance between the two. It is particularly useful when the class distribution is imbalanced. It is given by:

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

- **ROC Curve and AUC**

  The ROC curve is a graphical representation of a model's ability to distinguish between classes at various thresholds. It plots the True Positive Rate (Recall) against the False Positive Rate (1 - Specificity). The AUC, or Area Under the Curve, measures the entire two-dimensional area underneath the entire ROC curve and provides an aggregate measure of the model's performance across all possible classification thresholds. An AUC of 0.5 suggests that there is no discriminative ability (equivalent to random guessing), while an AUC of 1 indicates perfect classification.

Each of these metrics provides valuable insights into different aspects of a model's performance, and the choice of metric should align with the specific objectives and context of the ML problem being addressed.

2.2.4 Deep Learning

Deep learning, a significant breakthrough in the field of artificial intelligence (AI), has emerged as a transformative technology, reshaping how we interact with data and machines. Rooted in the principles of machine learning, deep learning extends these concepts by leveraging complex structures known as neural networks. These networks, inspired by the biological neural networks of the human brain, enable machines to process, analyze, and make decisions from large sets of data [73].

Central to deep learning is the architecture of Deep Neural Networks (DNNs), which distinguishes it from traditional machine learning. DNNs consist of multiple layers of interconnected nodes or neurons, where each layer performs specific computations on input data and

passes the results to subsequent layers [74]. This layered structure allows DNNs to learn from data in a hierarchical manner, extracting increasingly complex features at each level.

One of the most groundbreaking aspects of deep learning is its capability for feature extraction and pattern recognition. Traditional machine learning techniques rely heavily on manual feature extraction, which can be labor-intensive and requires domain expertise. In contrast, deep learning algorithms autonomously learn to identify relevant features directly from the data, a process that has significantly improved the efficiency and accuracy of various AI applications [75].

Deep learning has been instrumental in advancing fields such as computer vision and natural language processing (NLP). In computer vision, Convolutional Neural Networks (CNNs) have become the standard for tasks like image classification and object detection, providing the backbone for technologies ranging from facial recognition systems to autonomous vehicles [76]. Similarly, in NLP, models like Transformers have revolutionized language understanding and generation, leading to the development of highly sophisticated language models [77].

The training of deep learning models is a data and computationally intensive process. It involves feeding large volumes of data through the neural network and adjusting the network's parameters through backpropagation and optimization algorithms like stochastic gradient descent [78]. The ability of these models to learn from vast datasets has led to their unparalleled performance in tasks that were once considered challenging for machines.

Despite its successes, deep learning faces challenges such as the need for extensive data, the risk of overfitting, and the "black box" nature of its decision-making process, which can lack transparency [79]. Moreover, concerns regarding the ethical implications and biases in AI models pose significant challenges for the field [80].

As research continues to evolve, addressing its limitations and ethical considerations, deep learning stands poised to drive further innovations across various sectors, from healthcare to autonomous systems [81].

2.2.5 Deep Neural Networks (DNNs)

Deep Neural Networks (DNNs) are at the heart of the current revolution in AI and machine learning. As an advanced form of neural networks, DNNs are characterized by their depth, comprising multiple hidden layers between the input and output layers. This depth allows DNNs to model complex relationships in data, leading to groundbreaking performances in various AI tasks [82].

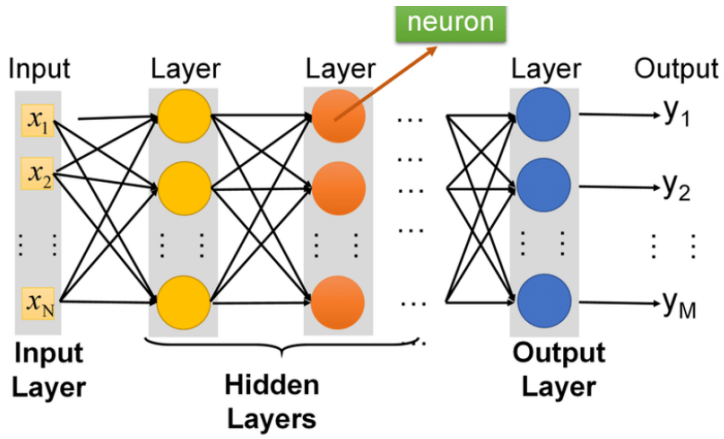

Figure 2.3: The Illustration of the Typical Architecture of a Deep Neural Network (DNN), Showcasing Multiple Hidden Layers, Neuron Connectivity, and Data Flow from Input to Output Layer [6].

A deep neural network (DNN) is composed of layers of nodes, also known as neurons, where each layer is specifically designed to carry out particular computations as shown in Figure 2.3. The input layer receives the raw data, which is then processed through successive hidden layers. Each neuron in these layers applies a set of weights to its inputs, followed by a non-linear transformation, often using activation functions like ReLU or sigmoid. The output layer produces the final result, be it a classification, regression, or any other desired output [83]. Training a DNN is a meticulous process, where the network learns by adjusting the weights of connections between neurons. This learning is achieved through backpropagation, a method that calculates the gradient of the loss function with respect to each weight by the chain rule, updating the weights to minimize the loss. Optimizers like stochastic gradient descent or Adam are commonly used to facilitate this process [84].

DNNs have made significant impacts across various domains. In computer vision, they enable advanced image recognition, object detection, and video analysis. In natural language processing, they have improved machine translation, sentiment analysis, and language generation. DNNs are also pivotal in autonomous systems, including self-driving cars and drones, where they process and interpret vast amounts of sensory data [85].

One of the critical challenges in working with DNNs is the requirement for large datasets and extensive computational resources. DNNs' performance improves with the amount and diversity of data they are trained on, necessitating powerful hardware, often GPUs or TPUs, for efficient training[86]. Additionally, the "black box" nature of DNNs, where the decision-making process is not always transparent, raises concerns, especially in critical applications like

healthcare or criminal justice [87].

Despite these challenges, the field of DNNs is evolving rapidly. Techniques like transfer learning, where a model trained on one task is repurposed on a second related task, have made it easier to deploy DNNs in environments with limited data. Furthermore, research into making DNNs more interpretable and fair continues to be a significant focus, aiming to build trust and reliability in AI systems [88].

2.2.6 Convolutional Neural Networks (CNNs)

Convolutional Neural Networks (CNNs) represent a groundbreaking advancement in the field of deep learning, particularly tailored for processing data that can be visualized in a grid-like structure, such as images. Unlike traditional neural networks, which flatten input data into a single vector, CNNs maintain the spatial hierarchy of the input, enabling the model to effectively capture the relationships within the data. This attribute makes CNNs exceptionally suited for image and video recognition tasks, where understanding the spatial context of pixels is crucial. The unique architecture of CNNs allows them to automatically and adaptively learn spatial hierarchies of features from images, a feature that has revolutionized the way machines understand visual content [89].

The architecture of a CNN is composed of several layers that transform the input image to produce the desired output, such as a class label for classification tasks as shown in Figure 2.4. The convolutional layer, the cornerstone of the CNN, applies numerous filters to the input to activate specific features at different spatial locations. Following the convolutional layer is the ReLU (Rectified Linear Unit) layer, which introduces non-linearity into the model by applying the function f(x) = max(0,x) to each pixel. This non-linearity allows the network to handle complex patterns and data structures. Subsequent pooling (or subsampling) layers serve to reduce the spatial dimensions of the input volume for the next convolutional layer, decreasing the computational burden and memory usage while preserving important features. The process within a CNN begins with the raw input image and progresses through feature learning and abstraction phases. Initially, convolutional and pooling layers work in tandem to detect simple features such as edges and gradients. As the data progresses deeper into the network, subsequent layers are able to recognize more complex features by combining the simpler patterns detected by earlier layers. This hierarchical approach to feature learning is pivotal in enabling CNNs to break down complex visual cues into manageable parts, facilitating the recognition of larger and more complex objects and patterns in later stages [90].

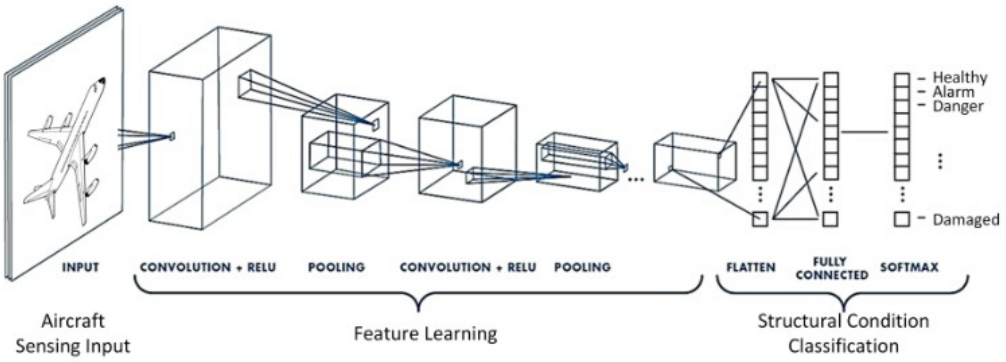

Figure 2.4: An illustrative diagram of a Convolutional Neural Network (CNN), showcasing the sequence of convolutional, ReLU, pooling, and fully connected layers that enable the model to process and classify visual data with high accuracy [7].

Following the feature extraction and abstraction process, the high-level reasoning within the CNN is executed through fully connected layers. These layers connect every neuron in one layer to every neuron in the next layer, thus integrating the learned features into a format suitable for classification. The final layer in a CNN typically consists of a softmax activation function that outputs a probability distribution over the classes, based on the features identified in the image. This structured approach allows CNNs to achieve remarkable accuracy in tasks such as image and video classification, showcasing their capability to not only recognize but also interpret complex visual information [91].

CNNs have found applications across a broad spectrum of fields, ranging from image and video recognition to medical image analysis and natural language processing. Their ability to learn and recognize patterns from visual data has paved the way for innovations in autonomous vehicles, facial recognition software, and even in the analysis of medical images for diagnostics. The adaptability and efficiency of CNNs in handling spatial data make them a cornerstone technology in the ongoing exploration and development within the realm of artificial intelligence and machine learning [92].

2.2.7 The Emergence of ML in Avionic Systems

As ML begins to permeate the field of avionics, the traditional landscape of system assurance undergoes a significant transformation. The emergence of ML in avionics heralds a new era in the field of aerospace engineering, fundamentally redefining the capabilities and potential of avionic systems. The integration of ML in avionics, as outlined in recent studies [93], [94], involves leveraging data-driven algorithms to enhance various aspects of aircraft operation, from

flight safety to predictive maintenance unlike conventional deterministic systems, ML-based systems in avionics introduce a layer of probabilistic decision making, which poses new challenges and considerations for system assurance. MLSs in avionics are primarily designed to analyze and interpret vast streams of data from numerous onboard sensors and systems, enabling a more nuanced understanding of the aircraft's operational status. As noted by Helgo et al.[95], the ability to detect potential failures early and take proactive measures to reduce risks is crucial in predictive maintenance. This capability greatly improves aircraft safety and operational reliability. Furthermore, ML's role in optimizing flight paths and air traffic management is highlighted in the research by Aghdam et al.[96], demonstrating how these algorithms can process complex flight data to suggest more efficient routes, thereby reducing fuel consumption and minimizing environmental impact. This optimization extends beyond mere operational efficiency; it also plays a crucial role in managing increasingly congested airspaces, enhancing the overall safety and punctuality of air travel.

The advancement towards autonomous flight systems represents another frontier in the application of ML in avionics. These autonomous systems, equipped with sophisticated ML algorithms, can make real-time decisions, adapt to changing flight conditions, and potentially handle unforeseen scenarios with greater accuracy than traditional systems. The prospect of fully autonomous drones and, eventually, passenger aircraft, while still in its nascent stages, is rapidly becoming a tangible reality, thanks in part to the advancements in ML technologies.

However, the integration of ML into avionic systems is not without its challenges. One of the primary concerns, as pointed out by Jiang et al. [97], is ensuring the reliability and interpretability of ML models, particularly in high-stakes scenarios where safety is paramount. The aviation industry's stringent regulatory environment also presents significant hurdles, necessitating comprehensive testing and certification processes for ML-based systems [98].

2.2.8 Assurance and Machine Learning Models

The incorporation of ML models into avionics systems represents a significant shift in the methodologies used to assure these systems' reliability and safety. This integration challenges traditional assurance practices, as it introduces systems that are inherently less predictable and whose decision-making processes are not always easily interpretable. Clainche et al. in their research [99] emphasize the critical need for predictability and explainability in ML models deployed within the intricate operational environment of an aircraft.

In aviation, a domain where system failures can lead to catastrophic consequences, it's cru-

cial that ML models operate reliably and as intended across a broad spectrum of operational scenarios. This necessitates an exhaustive approach encompassing thorough testing, meticulous validation, and continuous monitoring of these models. Denney et al. [100] stress that this approach should extend beyond assessing initial model accuracy. It should also scrutinize the model's capability to sustain performance over time and across diverse operational conditions. This calls for rigorous testing protocols that replicate an extensive range of operational scenarios, alongside validation processes that thoroughly examine the model's decision-making pathways for consistency and rationality.

Continuous monitoring is pivotal in promptly identifying and addressing any performance drifts or degradation that might manifest in the dynamic operational environment of avionics systems. The robustness of ML models in avionics, therefore, is a multifaceted challenge. It demands persistent vigilance to maintain aviation's inherent high safety standards.

A critical component of this assurance process is the models' ability to effectively detect and respond to OOD instances. The significance of OOD detection lies in the fact that ML models are typically trained on specific datasets and may not inherently perform accurately or reliably when encountering data that significantly diverges from their training sets. In avionics, where rapid and unpredictable changes are common, the ability of decision-making systems to accurately handle such deviations is crucial for safety [101]. Addressing OOD instances involves implementing mechanisms that can recognize when the model is confronted with unfamiliar data, enabling it to either adapt its responses or defer to human oversight.

This proactive approach to model assurance is integral in maintaining the integrity and trustworthiness of ML-based systems in critical aviation applications. It ensures not only the functional reliability of these systems but also fortifies the confidence in their deployment in high-stakes environments.

2.2.9 Out-of-Distribution (OOD) Detection

In the increasingly sophisticated domain of avionics, the deployment of Machine Learning (ML) models introduces a critical challenge: the management of Out-of-Distribution (OOD) instances. These instances occur when the input data significantly deviates from the model's training data. They present scenarios that the model may not have encountered during its training phase. This concept is illustrated in Figure 1.2, which might show a graph comparing model performance on training data versus novel or OOD data [102].

OOD parameters are crucial to understand in this context. For example, consider an ML

model trained on data from commercial flights under typical weather conditions. An OOD instance might occur if the model encounters data from severe weather conditions like a sudden thunderstorm or a volcanic ash cloud - situations it was not trained on. The ML model's ability to recognize and respond to such OOD instances, as Liu et al. have elucidated, is vital for maintaining the reliability and safety of avionics systems [103].

In the field of aviation, where safety is of utmost importance, the objective is to minimize uncertainty within the system, even though the operating environment is inherently subject to variability [104]. This is especially relevant when introducing ML into avionics systems. The introduction of ML should not introduce unpredictability; rather, it should enhance the system's ability to handle the inherent variability and unforeseen events in aviation with greater precision and reliability [105].

Given this context, the role of effective ODD detection becomes crucial. OOD detection in ML-based system is designed to identify and appropriately respond to data or situations that deviate significantly from the model's training data [106]. In the highly controlled and regulated field of aviation, where every component and system must perform predictably and reliably, the integration of ML must adhere to these stringent standards [107]. ML models in avionics systems must be rigorously tested and validated to ensure they can not only handle expected scenarios but also identify and adapt to unforeseen conditions without introducing additional unpredictability into the system [108].

The challenge lies in ensuring that the ML models contribute to the predictability and robustness of the avionics systems, rather than detracting from it. This involves a careful balance of leveraging the advanced capabilities of ML to enhance system performance, while also rigorously assessing and mitigating any potential sources of unpredictability that these models might introduce [109]. Ensuring this balance is essential for the safe and reliable operation of ML-integrated avionics systems in the dynamic and often unpredictable aviation environment. While ML models are robust within their learned contexts, their accuracy and reliability can decrease when confronted with data that falls outside of their training distribution. In avionics, where the stakes are exceptionally high, any error in decision-making by the ML model can lead to serious consequences [110].

Therefore, the implementation of advanced OOD detection mechanisms is imperative. These mechanisms function by continuously monitoring input data and comparing it to the known distribution of the model, effectively flagging data points that lie outside this distribution. For instance, if an ML model used for flight path optimization encounters unexpected air traffic patterns due to an unscheduled event, an effective OOD system would recognize this deviation

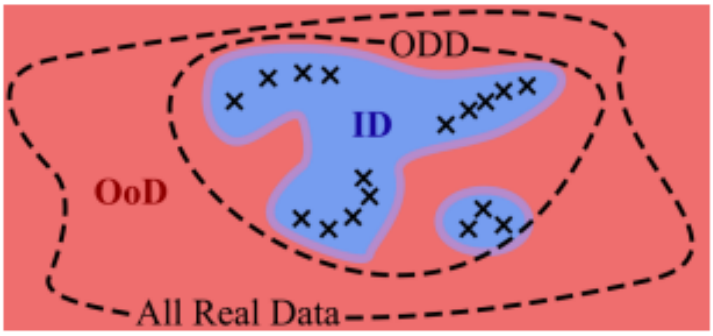

Figure 2.5: The illustration depicts the distribution of input data for the model, with crosses indicating individual data points. The blue area denotes the In-Distribution (ID) region, where data aligns with the model training, while the red area highlights the Out-of-Distribution (OOD) region, representing data that deviates from the model training set [8].

from normal traffic patterns [111]. Upon detecting an OOD instance, these mechanisms can initiate appropriate responses. These responses may involve reverting to human oversight or activating alternative decision-making protocols designed to handle such anomalies.

The development and integration of robust OOD detection systems are pivotal in ensuring that ML models in avionics remain reliable and safe, even when faced with unexpected or novel data inputs. By doing so, these systems safeguard the overall integrity of the aviation system, ensuring that ML-enhanced avionics can adapt to and manage the complexities of real-world flight environments

2.2.10 Approaches for Out-of-Distribution Detection

- **Maximum Softmax Probability (MSP)** : The method is a straightforward yet effective approach to Out-of-Distribution (OOD) detection in neural networks. This technique was introduced by Hendrycks and Gimpel in their seminal 2016 paper [112]. The core idea behind MSP is to utilize the softmax output of a neural network as a confidence measure to distinguish between in-distribution (ID) and OOD samples. In practical terms, MSP examines the maximum value of the softmax probabilities produced by a network for a given input. The underlying hypothesis is that a network is more likely to produce higher maximum softmax probabilities for ID data, which it has encountered during training, compared to OOD data, which is unfamiliar to the network.

  The effectiveness of MSP lies in its simplicity and ease of integration with existing models. Since it relies on the final softmax layer, which is common in classification networks, MSP can be applied without the need for architectural changes or additional training [113]. This aspect makes it particularly appealing for real-time applications where computational efficiency is crucial.

However, MSP is not without limitations. Its performance can vary depending on the nature of the data and the model. For instance, in scenarios where OOD samples are not sufficiently different from ID samples, MSP might not provide a clear distinction, leading to potential false positives or negatives [114].

- **Multi-layer Out of Distribution (OOD) Detection**: is an advanced approach in the field of machine learning that involves utilizing features from multiple layers of a Deep Neural Network (DNN) to enhance the detection of OOD samples [115]. Traditional OOD detection methods often rely on the output layer's softmax probabilities, but multi-layer techniques leverage the rich, hierarchical representations learned at different levels of the network. By analyzing the activations from various layers, this method can capture a broader range of anomalies that might not be evident at the final output layer alone.

  One significant advantage of multi-layer OOD detection is its ability to exploit the diverse feature representations inherent in deep networks. Early layers might capture basic patterns and textures, while deeper layers encapsulate more complex, abstract representations [116]. This multi-level analysis allows for a more nuanced and robust identification of OOD samples, especially in cases where anomalies are subtle or deeply embedded in the data.

  However, there are limitations to this approach. One key challenge is the increased computational complexity. Analyzing activations from multiple layers requires more processing power and memory, which can be a bottleneck in resource-constrained environments or real-time applications [117]. Another issue is the potential for overfitting to the idiosyncrasies of the training data. If the model is too finely tuned to the specific characteristics of the training set, it may struggle to generalize effectively to new, unseen OOD samples.

  The effectiveness and limitations of multi-layer OOD detection are discussed in various research papers. For instance, in [118] their work on a unified approach to detecting both adversarial and OOD examples, highlighted the potential of using features from multiple network layers. They demonstrated improved detection performance but also acknowledged the need for careful calibration and the potential risk of increased false positives.

- **ODIN (Out-of-Distribution detector for Neural networks)**: is a specialized technique for enhancing the detection of out-of-distribution (OOD) samples in deep learning models, particularly in neural networks. Introduced by Henriksson et al. in their 2017 paper [119], this method is notable for its ability to improve OOD detection without needing any modification to the architecture or retraining of the pre-existing neural network models.

The core idea behind ODIN is to utilize temperature scaling and input preprocessing to effectively separate the softmax score distributions of in-distribution (ID) and OOD samples. Temperature scaling involves adjusting the temperature parameter in the softmax function, which helps in amplifying the differences in the predicted class probabilities [120]. This scaling makes the network's output more confident for ID samples and less confident for OOD samples. Input preprocessing, on the other hand, involves adding small perturbations to the input data. These perturbations are designed in a way that they increase the model's maximum softmax probability for ID samples, thereby creating a larger gap between ID and OOD samples [121].

However, ODIN is not without limitations. One significant challenge is the fine-tuning of hyperparameters, such as the temperature scaling factor and the magnitude of input perturbations, which are crucial for the effectiveness of the method. These parameters often require careful calibration and may not generalize well across different datasets or neural network architectures [122]. Furthermore, the reliance on softmax scores means that ODIN might still struggle in cases where the neural network is overconfident in its predictions, a common issue in deep learning models.

- **Channel-Attention-based Neural Mean Discrepancy (CA-NMD)**: is an advanced method for Out-of-Distribution (OOD) detection that combines the concepts of channel attention mechanisms with neural mean discrepancy analysis [123]. This approach is designed to enhance the sensitivity of neural networks to OOD samples by focusing on the most informative features in the data.

  In CA-NMD, the channel attention mechanism is employed to weigh the importance of different channels in a convolutional neural network (CNN) [124]. This mechanism adaptively adjusts the contribution of each channel based on its relevance to the task, which is particularly useful in complex datasets where certain features are more indicative of OOD samples than others. By emphasizing these relevant features, the network becomes more proficient in distinguishing between in-distribution and OOD data.

  The neural mean discrepancy component of CA-NMD involves computing a statistical measure that quantifies the difference between the distributions of in-distribution and OOD samples [125]. This is typically achieved by assessing the mean feature representations of the two distributions. A larger discrepancy suggests that the sample is likely to be OOD.

  One limitation of the CA-NMD approach is its potential computational overhead. The integration of channel attention mechanisms and mean discrepancy calculations can increase the complexity of the model, potentially leading to higher computational costs and longer inference times [126]. This makes it less suitable for applications where real-time detection and low latency are critical.

  Furthermore, like many OOD detection methods, CA-NMD may face challenges in scenarios where the OOD samples are very similar to the in-distribution data. In such

cases, the channel attention mechanism might not effectively discriminate between the subtle differences of the distributions, leading to decreased detection performance [127].

- **The Out-of-Distribution (OOD) Discernment Layer (OODL)**: is a specialized neural network layer designed to improve the detection of out-of-distribution data, a crucial aspect in ensuring the reliability and safety of machine learning models [128]. This approach involves adding an extra layer to a pre-existing neural network model, trained explicitly to distinguish between in-distribution (ID) and OOD samples. The fundamental idea is to enhance the model's sensitivity to data that deviates from the training distribution, thereby mitigating risks associated with unexpected inputs during deployment [129].

  In practice, the discernment layer learns to identify patterns and features that are characteristic of OOD samples [130]. This is typically achieved by exposing the layer to both ID and OOD data during the training process, allowing it to learn a decision boundary that effectively separates the two. The discernment layer's output can then be used as an indicator of whether a given input is likely to be OOD, providing an additional layer of safety in critical applications.

  However, this approach is not without its limitations. One significant challenge is the requirement for representative OOD data during training. The effectiveness of the discernment layer heavily depends on the diversity and relevance of the OOD samples used in training [131]. If the OOD samples are not representative of the potential deviations encountered in real-world scenarios, the layer may fail to detect certain types of OOD inputs. Additionally, the integration of an additional layer into a neural network can introduce computational overhead, potentially impacting the model's efficiency and latency, which can be a concern in resource-constrained environments or real-time applications.

  Another limitation arises from the potential overfitting to the OOD samples used during training. If the layer becomes too specialized in recognizing the specific OOD samples it was trained on, it might not generalize well to other types of OOD data encountered post-deployment [132]. This specialization can limit the model's robustness and adaptability to new, unseen OOD instances.

  Despite these challenges, the Out-of-Distribution Discernment Layer presents a valuable approach for enhancing the robustness of machine learning models, especially in scenarios where encountering OOD data is likely and can have significant consequences [133]. Ongoing research in this area focuses on addressing these limitations, aiming to develop more generalized and efficient OOD detection mechanisms.

- **Mahalanobis** :The Mahalanobis distance-based method for Out-of-Distribution (OOD) detection in machine learning leverages the statistical properties of the feature space to differentiate between in-distribution (ID) and OOD samples. Introduced by Lee et al [134], this approach calculates the Mahalanobis distance of a test sample from the

means of the class-conditional distributions learned during the training of a Deep Neural Network (DNN). The Mahalanobis distance, a measure rooted in statistics, effectively captures how far a given sample deviates from the expected distribution of the trained classes.

To implement this, feature vectors are extracted from various layers of the DNN, and the Mahalanobis distance is computed for each class. These distances are then used to generate a confidence score indicating the likelihood of a sample being OOD [135]. The method can be further enhanced by incorporating input preprocessing techniques and logistic regression-based post-processing to improve detection performance. One significant limitation of the Mahalanobis method is its dependency on the assumption that the data in feature space follows a Gaussian distribution [136]. This assumption might not hold in complex, real-world datasets where the distribution can be highly non-linear and multi-modal. Additionally, the performance of this method can degrade if the feature extraction layers of the DNN are not well-aligned with the statistical properties assumed by the Mahalanobis distance calculation. This misalignment can lead to less effective differentiation between ID and OOD samples, particularly in scenarios where the OOD samples are not drastically different from the ID samples [137].

- **Energy-based Out-of-Distribution (OOD) Detection**: Energy-based OOD detection is a relatively recent method that uses the concept of energy scores derived from a neural network's outputs to distinguish in-distribution (ID) from out-of-distribution samples. This method was introduced by Liu et al [138]. in their 2020 paper, where they proposed the idea of using an energy score as a more effective measure for OOD detection compared to traditional softmax probabilities. The energy score is computed as a function of the network's logits, and it offers a scalar quantity that is lower for ID samples and higher for OOD samples. The intuition behind this method is that ID data should align well with the learned energy landscape of the model, while OOD data would not[139].

  The key advantage of energy-based OOD detection lies in its simplicity and effectiveness. Unlike some methods that require extensive modification to the network or additional training, this approach can be applied directly to pre-trained models. Furthermore, it addresses some limitations of softmax-based methods, which can be overly confident about OOD samples [140].

  A notable limitation of the energy-based OOD detection method is its potential sensitivity to the hyperparameters used for computing the energy score. The effectiveness of this method can vary depending on the choice of these parameters, which may require careful tuning [141]. Additionally, while it provides an improvement over softmax-based methods, it may not always outperform other sophisticated OOD detection techniques, especially in scenarios with subtle differences between ID and OOD samples.

- **Softmax Response for Out-of-Distribution (OOD) Detection** :The Softmax Response method is a straightforward approach for Out-of-Distribution (OOD) detection in neu-

ral networks [142]. It relies on the final softmax layer of a classifier, which provides a probability distribution over the predicted classes. In this method, the confidence of the network's predictions is assessed by examining the softmax output [143]. Typically, a high maximum softmax probability is interpreted as the network being confident about its prediction, suggesting that the input is likely in-distribution (ID). Conversely, a lower maximum probability indicates less confidence, potentially flagging the input as OOD. One significant limitation of the Softmax Response method is its inherent assumption that high confidence equates to the correctness of a prediction, which is not always true [144]. Neural networks, especially deep ones, are often overconfident in their predictions, even when dealing with OOD samples. This overconfidence leads to scenarios where an OOD sample is assigned a high softmax score, falsely indicating that it is in-distribution. This limitation is particularly pronounced in cases where the OOD samples are not drastically different from the ID data or when the network is exposed to adversarial examples.

Moreover, the softmax function can be overly simplistic for complex OOD detection tasks. It does not account for the multi-dimensional nature of feature spaces in deep learning models, potentially overlooking subtle yet critical differences between ID and OOD samples. This issue is exacerbated in high-dimensional spaces where softmax probabilities can become less discriminative [145].

2.2.11 TinyML

TinyML, short for Tiny Machine Learning, is an emerging field in technology that focuses on developing machine learning models that are small enough to run on low-power hardware devices [146], such as microcontrollers as shown in Figure 2.6. This field is gaining traction because it allows for the deployment of advanced machine learning algorithms in extremely small and power-efficient devices, often with limited computational resources and memory. TinyML is particularly significant in the realm of Internet of Things (IoT) and embedded systems, enabling smart functionalities in a wide range of devices, from wearable sensors to home appliances.

TinyML has made significant progress in the area of embedded control systems. These systems are integral to many electronic devices and machinery, controlling specific functions within these systems. With TinyML, these embedded systems can now incorporate intelligent features like predictive maintenance, voice recognition, and anomaly detection, all while operating independently from larger, more power-intensive computing resources. This advancement

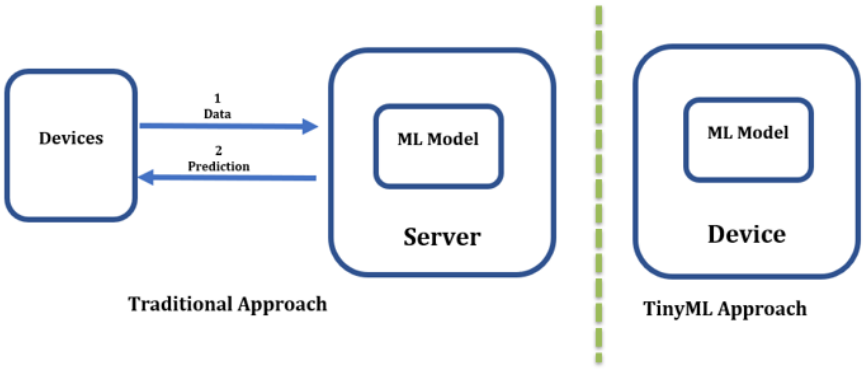

Figure 2.6: The figure illustrates Comparative Analysis of Traditional Software Development Approach versus Tiny Machine Learning

allows for more responsive, efficient, and smart control systems in various applications, from industrial machines to consumer electronics [147].

The adoption of TinyML in safety-critical systems is an area of growing interest but also comes with stringent requirements due to the high stakes involved [148]. Safety-critical systems are those where failures could result in significant harm or loss of life, such as in medical devices, automotive systems, or aviation. In these domains, the reliability, robustness, and explainability of ML models are paramount. TinyML's role in these systems is still evolving. The primary challenge lies in ensuring that the models are not only accurate and efficient but also fail-safe and interpretable. In safety-critical systems, any ML-based decision or action must be highly reliable and, ideally, understandable to human operators or engineers. Therefore, while there are potential applications of TinyML in safety-critical systems, its implementation is cautious and heavily scrutinized.

When it comes to OOD detection, TinyML faces unique challenges. OOD detection is the process of identifying data that is significantly different from the training dataset of a model [149]. In the context of TinyML, the limited computational power and memory capacity of devices make the implementation of robust OOD detection mechanisms more challenging. However, effective OOD detection is crucial, especially in safety-critical applications, to ensure that the ML model behaves reliably under all circumstances.

Developing efficient algorithms that can detect OOD instances without overburdening the limited resources of TinyML devices is an area of active research [150]. This involves creating lightweight yet effective models that can recognize when incoming data does not match the trained scenarios, prompting appropriate system responses [147].

2.2.12 TinyML for Identifying Out-of-Distribution Instances

TinyML is an innovative advancement in the field of machine learning, particularly in the realm of machine learning systems that prioritize size, power, and efficiency. This approach is centered around the deployment of compact machine learning models on small, low-energy devices, like microcontrollers and edge devices. The essence of TinyML lies in its ability to bring the power of machine learning to the smallest of devices, enabling intelligent decision-making capabilities in a wide range of applications, from wearable health monitors to industrial sensors and beyond [30]. Its compact nature allows for on-device processing, reducing the need for constant connectivity and data transmission to centralized servers, thereby enhancing data privacy and reducing latency. TinyML plays a transformative role in MLSs assurance, particularly where efficiency, size, and real-time response are critical. By enabling ML models to run on low-power, small-scale devices, TinyML ensures continuous and reliable operation in environments where traditional, larger ML-based system are impractical. This is crucial for applications requiring long-term, autonomous functioning without frequent maintenance or power replenishment, such as in remote environmental sensors or wearable health devices [151].
The advent of TinyML marks a significant advancement in the field of avionics, addressing some of the key challenges associated with integrating ML into such systems. TinyML, as defined by Warden et al. in [30], involves the deployment of compact ML models on low-power hardware, a solution perfectly suited to the unique demands of avionics systems. These models are designed to be lightweight yet capable of performing complex computations, a necessity given the limited power and computational resources available on aircraft [152].

One of the main advantages of TinyML in avionics is its ability to facilitate real-time processing and decision-making. This capability is particularly crucial for critical tasks such as Out-of-Distribution (OOD) detection, where the system must quickly identify and respond to data points that deviate from the norm. In scenarios where every second counts, such as in-flight decision making or emergency response, the rapid processing power of TinyML can be invaluable [153].

The integration of TinyML for OOD detection is a game changer in several respects. First, TinyML models are designed to operate on low-power, resource-constrained devices, allowing them to be embedded directly into a wide range of systems. This embedded nature enables real-time data processing and immediate OOD detection, which is essential for timely responses in critical situations. For example, in avionic systems, TinyML can continuously monitor sensor data, quickly identifying anomalies that might indicate a potential system malfunction or an

unexpected environmental condition [154].

Furthermore, the lightweight architecture of TinyML models makes them ideal for continuous monitoring on-device without draining significant power resources [155]. This continuous monitoring is key for early OOD detection, ensuring that deviations from normal operating conditions are identified and addressed promptly, thereby maintaining the system's integrity and safety.

Another significant advantage of TinyML in OOD detection is its ability to operate independently of network connectivity [156]. This is particularly important in scenarios where consistent connectivity cannot be guaranteed. By processing data locally, TinyML models can effectively detect OOD instances even in offline modes, ensuring uninterrupted safety monitoring.

2.3 W-shaped Development Cycle for the Assurance of Avionics systems

The integration of ML into avionics systems necessitates a robust and comprehensive development cycle, for which the W-shaped model as shown in Figure 2.7 is particularly well-suited. This model, an adaptation of the traditional V-model, introduces an additional layer of validation and verification, forming a 'W' shape, and is crucial in addressing the unique challenges posed by ML in avionics [9]. The W-shaped development cycle begins with an in-depth analysis of system requirements, a stage crucial for understanding the operational needs and constraints specific to avionics systems and the role of ML within them. This is followed by the system design phase, where high-level architecture is established, considering both traditional avionic components and ML elements, as highlighted by Johnson and Smith in [157]. The subsequent phase involves the component design and development, where individual parts of the system, including ML models, are developed. Each component, particularly the ML algorithms, is rigorously tested to ensure they meet specified requirements, addressing both functional and safety aspects, as emphasized in [158] the work by Taylor et al. This is the first downward stroke of the 'W'.

The next stage involves integrating these components and conducting comprehensive system testing (the first upward stroke), ensuring all parts, including ML models, function harmoniously. Following this, the system undergoes validation (the second downward stroke), where it is ensured that the system meets the operational needs and complies with stringent

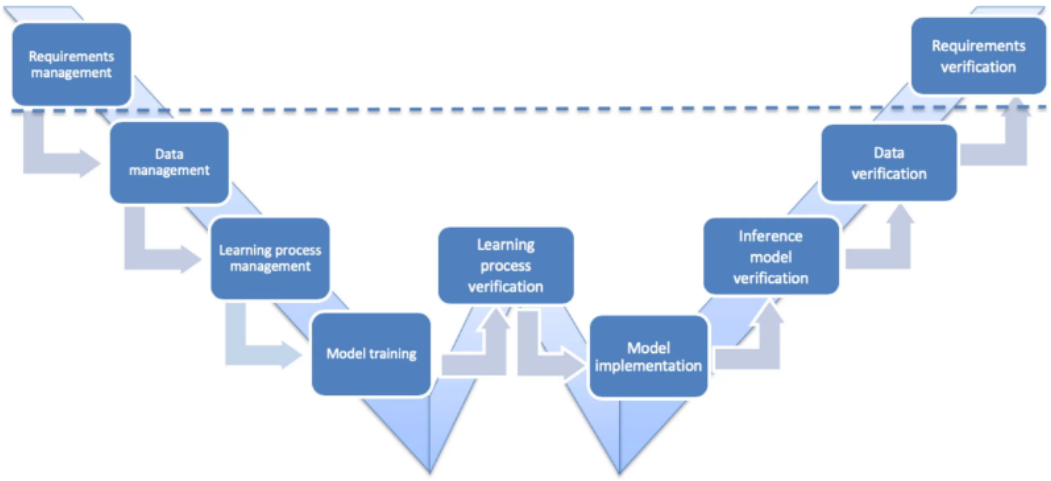

Figure 2.7: The diagram illustrates the W-shaped Learning Model, a methodology proposed by EASA and Daedalean. This model encapsulates the core principles of the learning assurance life cycle, designed to guide the development of machine learning applications that meet stringent requirements. It emphasizes a structured, comprehensive approach to ensuring the reliability and efficacy of MLSs in critical applications [9]. (2020)

avionics regulations, a process detailed in [159] by Torens et al.

Finally, the system enters the operation and maintenance phase (the second upward stroke), where it is subject to continuous monitoring and periodic updates. This phase is crucial for ML models, as it involves ongoing learning and adaptation to new data and operational conditions, ensuring sustained reliability and safety in dynamic avionic environments. Comparing the classical V-model for system development to the adapted W-model for neural network integration, we see that the requirements gathering and verification remain consistent with traditional approaches. This is the portion above the dotted line, where the focus is on defining and verifying the system requirements, a process well-established in existing standards [160].

Below the dotted line, the W-model introduces a new realm of considerations specific to neural networks. Ensuring that these ML model operate as intended involves a trio of critical elements: the design of the data, the learning process, and the model itself. Each of these elements is essential and can be a potential point of failure if not handled correctly. Let's delve into how each of these elements is addressed in the W-model.

- **Correctly Designed Data** :
  - Data Collection: Ensuring a comprehensive and representative dataset that covers the expected operational scenarios of the neural network.
  - Data Preprocessing: Cleaning, normalizing, and structuring data in a way that opti-

mizes the learning process.
- Data Validation: Verifying the quality and relevance of the data to the intended application of the neural network.

- **Correctly Designed Learning Process**:
- Algorithm Selection: Choosing the right learning algorithm based on the problem context and the nature of the data.
- Parameter Tuning: Adjusting the learning parameters to optimize the performance of the neural network.
- Training Methodology: Implementing effective training procedures to ensure robust learning without overfitting.

- **Correctly Designed Model**:
- Architecture Design: Structuring the neural network architecture to suit the complexity and nature of the task.
- Performance Evaluation: Assessing the model's performance through various metrics to ensure it meets the desired criteria.
- Robustness and Generalization: Ensuring the model can handle new, unseen data and maintain performance across different conditions.

In the W-process, we "walk down" by delving into each of these three elements, thoroughly developing and refining them. As we "walk up," we validate and verify that each element aligns with the initial requirements and performs effectively in the intended operational environment. This comprehensive approach ensures that the ML model not only meets the specified requirements but is also robust, reliable, and capable of performing its intended function in the real world [161].

W-shaped development cycle provides a structured and rigorous framework for the integration of ML in avionics, addressing the entire spectrum from initial requirements to ongoing operation, a necessity for maintaining the high safety and reliability standards in the aviation industry [162].

2.4 Summary

The ML into avionics systems represents a significant evolution in aviation technology, poised to enhance the functionality and efficiency of these systems. However, this integration introduces a broader problem space, encompassing the intricacies of embedding ML within the

strict confines of safety-critical systems, specifically in avionics. This chapter seeks to address this expansive problem space, focusing on two principal elements: the unique attributes of safety-critical avionics systems and the nuances of incorporating ML into these systems' command-and-control segments.

One key aspect under scrutiny is the Operational Design Domain, which is becoming increasingly vital in the assurance of safety-critical systems. OOD's role in avionics is to enhance the precision and agility with which implementors can conform to stringent certification standards. This concept is essential for understanding how ML can be integrated into avionics systems while maintaining the highest safety standards.

The chapter further delves into the essence of ML and its growing impact on revolutionizing the avionics industry. It explores the challenges posed by the integration of ML into aviation systems, with a special emphasis on detecting Out-of-Distribution situations. OOD detection is critical because it involves identifying and managing data or scenarios that the ML model has not encountered during training, a common occurrence in the dynamic aviation environment. Addressing these challenges is crucial for ensuring the safety and reliability of ML-driven avionics systems.

Additionally, the chapter examines the potential role of TinyML in bridging the gap between innovation and adherence to the conservative safety regulations prevalent in aviation. TinyML offers a pathway to incorporate ML into avionics by balancing the innovative aspects of ML with the stringent safety requirements inherent to aviation.

Overall, this chapter provides a comprehensive overview of the larger space of integrating ML into avionics systems. It highlights the importance of understanding and addressing the unique challenges of this integration, from the perspective of both safety-critical system requirements and the specific demands of embedding ML models in avionics. The focus is on ensuring that the advancements brought by ML, such as enhanced decision-making and operational efficiencies, are realized within the framework of the fundamental safety requirements of aviation technology.

3

Objectives, Methodology, and Results

In this chapter, we dive into the practical methodologies and approaches implemented to address the research questions (RQ) outlined in Chapter 1. This exploration is structured through distinct sections:

- The first section presents a detailed overview of our research objectives, explicitly framed in the context of the three main research questions. Here, we elucidate how each objective aligns with and contributes to answering these questions, thereby setting a clear path for our investigative journey.
- The second section is dedicated to presenting the specifics of our experimental setup, including the objectives that guided its design. This section also comprehensively details the empirical results and findings obtained from the execution of these experiments, offering insights into their significance and implications.

3.1 Research Objectives

Building upon the research questions (RQs) outlined in Section 1.5 and the foundational understanding established in Chapter 2, we have identified specific research objectives. These objectives are presented below, accompanied by our reasoning, to illustrate how they align with the RQs and are informed by the background knowledge of the related concepts.

The first research question (RQ-1) was:
"What specific standards and conditions must be established for crafting training scenarios for Machine Learning models to guarantee thorough training and consistent, reliable performance across a wide range of practical applications, including control-loop systems in avionics?"
To establish optimal criteria and requirements for defining the training scenarios of Machine Learning models, ensuring comprehensive learning and robust performance across diverse real-world applications. This will be achieved by defining a detailed Operational Design Domain (ODD) that encompasses a wide range of variables and conditions under which the models are expected to operate. The ODD will serve as a framework to guide the selection and creation of training datasets and scenarios, ensuring that they are representative of the diverse and complex environments the models will encounter in practical applications. This approach aims to enhance the models' ability to generalize and perform effectively in a variety of situations, thus addressing the critical need for reliability and adaptability in machine learning applications. Thus, we define the following first objective, which is clearly highlighted in the W-shaped development life-cycle Figure 3.1.

1. **In-Distribution and Out-Distribution Operational Design Domain**:

    - **In-Distribution Operational Design Domain**: Establish criteria for in-distribution scenarios within the operational design domain, focusing on situations where the machine learning model is trained to recognize close-up images of airplanes. This objective includes specifying the range of distances, angles, lighting conditions, and backgrounds for which the model should be optimized, ensuring high accuracy and reliability in scenarios closely aligned with the training data.
    - **Out-of-Distribution Operational Design Domain**: Develop requirements for 'out-of-distribution' scenarios in the operational design domain, where the model encounters far-off images of airplanes, differing significantly from its training data. This will involve identifying and characterizing external factors, such as extreme distances, varied environmental conditions, and atypical perspectives, to train the model to maintain performance and robustness even when presented with data that fall outside its primary training distribution.

The second research question (RQ-2) was:
"What methodologies and measures can be implemented during the development phase of Machine Learning models in avionics to effectively minimize the risk of performance insufficiencies and ensure robust functionality when deployed in open-world environments?"

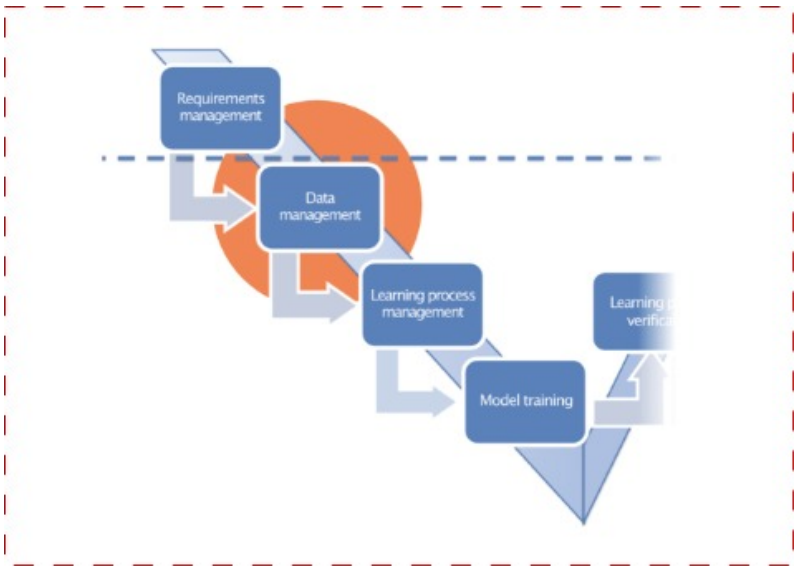

Figure 3.1: Part of the W-shaped Development Lifecycle (Figure 2.7) in a Machine Learning System, Focusing on Data Management for Operational Domain Design (ODD) and Out-of-Domain (OOD) Considerations.

To investigate and implement a multi-layer early exit strategy for out-of-distribution detection in machine learning models, particularly in the realm of avionics, enabled by TinyML technology. This objective focuses on developing methodologies and measures during the development phase that can preemptively identify and handle instances where the model encounters data significantly deviating from its training set, especially in open-world environments. The aim is to integrate layered decision-making points within the model architecture, allowing for early detection and response to out-of-distribution data, thereby minimizing risks of performance insufficiencies and ensuring robust and reliable functionality in diverse and unpredictable real-world scenarios. This approach seeks to leverage the compact and efficient nature of TinyML to provide a scalable and effective solution to improve the safety and dependability of machine learning applications in aviation. Hence our second objective was following which is highlighted in the W-shaped development life-cycle Figure 3.2:

2 **The Multi-Layer Framework-Fast TinyML OOD Detector (FTO)**: This objective encompasses the development of a comprehensive framework that showcases the use of early exit strategies within a Deep Neural Network (DNN), specifically tailored for efficient and seamless Out-Of-Distribution detection. The framework will detail the integration of multiple decision layers within the DNN architecture, allowing for prompt identification and handling of OOD data at various stages of processing. This multi-layer approach aims to enhance the model's responsiveness and accuracy in real-time scenarios, ensuring robust performance in diverse and challenging environments.

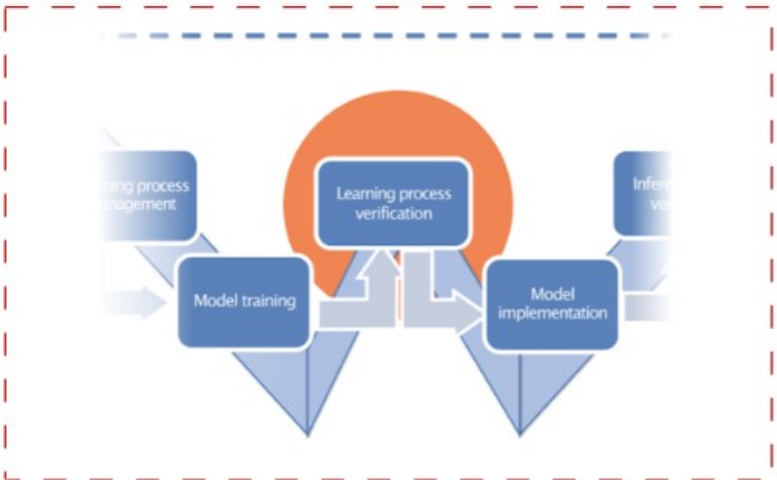

Figure 3.2: Diagram Illustrating the Segment of the W-Shaped Development Lifecycle (Figure 2.7) Featuring the Multi-Layer Framework-Fast TinyML Out-Of-Distribution (FTO) Detector for Learning Process Verification in a Machine Learning System

The third research question (RQ-3) was:
"How does embedding Machine Learning (ML) in avionic systems contribute to enhancing model assurance, particularly by boosting reliability, safety, and instant decision-making, despite the aviation industry's limited computational capacity and strict regulatory requirements?"

To explore the integration of a multi-layer early exit strategy for out-of-distribution detection in Deep Neural Networks, utilizing TinyML for enhanced outlier detection in one or more hidden layers, in the context of avionics embedded systems. This objective involves the development and implementation of a runtime monitoring system *(A runtime monitoring system for machine learning actively tracks and evaluates an ML model's performance during its deployment, ensuring it operates correctly and efficiently. It's especially crucial in critical systems like avionics control-loops, where it detects performance deviations, anomalies, and adjusts thresholds for errors in real-time to maintain safety and reliability)*, based on the Local Outlier Factor (LOF) algorithm. The aim is to improve the reliability, safety, and real-time decision-making capabilities of Machine Learning models in avionics, while efficiently operating within the constraints of limited computational resources and adhering to stringent regulatory standards. This objective focuses on how the synergy of TinyML lightweight framework and LOF-based runtime monitoring can offer a robust solution for detecting and managing atypical data, thereby significantly enhancing the assurance and operational integrity of ML models in the aviation industry.Our last objective is following , which is also highlighted in the W-shaped

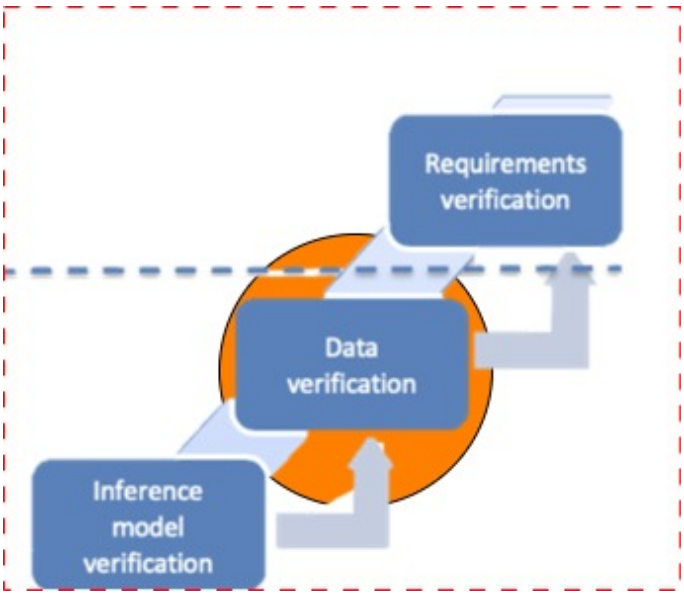

Figure 3.3: Diagram Illustrating the W-Shaped Development Lifecycle (Figure 2.7)featuring the Multi-Layer Early Exit for Out-of-Distribution (OOD) Detection with Local Outlier Factor (LOF) in Deep Neural Networks (DNNs) for Data Verification, Crucial for Ensuring Runtime Assurance in Machine Learning Systems

development life-cycle Figure 3.3:

3 **Implementing Multi-Layer Early Exit for OOD Detection with LOF in DNNs (MELOD)**: To implement a multi-layer early exit strategy in Deep Neural Networks (DNNs) for efficient out-of-distribution detection, empowered by TinyML. This involves integrating outlier detection mechanisms within one or more hidden layers of the DNN, using a runtime monitor based on the Local Outlier Factor (LOF). This approach aims to enhance the detection and management of atypical data in real-time, improving the overall reliability and safety of ML models in resource-constrained environments like avionics systems.

For each defined research objective, we conducted an empirical research study, employing a variety of methods to meticulously analyze and manipulate the data. These methods were chosen and customized to effectively address the specific aspects and requirements of each objective, ensuring a comprehensive and rigorous examination of the underlying hypotheses and questions.

3.2 Datasets Overview

Datasets are crucial in the advancement and verification of machine learning models, forming the basis for algorithms to acquire knowledge and progress. This section examines the various datasets employed in our research, each carefully selected to address particular elements of the Fast TinyML Out-of-Distribution (FTO) detection algorithm. We will investigate the features, origins, and reasons for choosing these datasets, aiming to provide a thorough insight into how they enhance the strength and effectiveness of our OOD detection techniques.

The following detailed explanations of the datasets cover their purpose, characteristics, and the context in which they are used for the model mentioned. Please note that while CIFAR-100 is listed both as a training (in-distribution) and evaluation (OOD) dataset, it is typically used in one context or the other. For clarity, I'll describe it in both contexts.

In-Distribution and Out-Distribution Datasets

1. **CIFAR-10**[163]

    - **Purpose:** Used for training and benchmarking machine learning models in image recognition tasks.
    - **Characteristics:** Contains 60,000 32x32 color images in 10 different classes, with 6,000 images per class. The dataset is divided into 50,000 training images and 10,000 test images.

2. **CIFAR-100**[163]

    - **Purpose:** Similar to CIFAR-10 but with more granularity, used for training models when a finer classification is required.
    - **Characteristics:** Includes 60,000 32x32 color images but divided into 100 classes, each with 600 images. Like CIFAR-10, it has a standard division of 50,000 training images and 10,000 test images.

Out-Distribution Test Datasets

1. **MNIST**[164]

    - **Purpose:** Benchmark dataset for handwritten digit recognition.
    - **Characteristics:** Comprises 70,000 28x28 grayscale images of 10 digit classes (0-9). It's typically split into 60,000 training and 10,000 test images.

2. **K-MNIST**[165]

    - **Purpose:** Designed as a more challenging, drop-in replacement for MNIST, featuring Japanese Kuzushiji characters.
    - **Characteristics:** Contains 70,000 28x28 grayscale images of 10 classes, each representing a different Japanese character.

3. **Fashion-MNIST**[166]

    - **Purpose:** Another drop-in replacement for MNIST, aimed at serving the same image classification tasks with a dataset of fashion articles.
    - **Characteristics:** Includes 70,000 28x28 grayscale images across 10 categories like trousers, dresses, bags, etc.

4. **LSUN (Large-scale Scene Understanding)**[167]

    - **Crop Purpose:** Used for scene understanding and recognition tasks, focusing on the central scene or object.
    - **Resize Purpose:** Adapted for compatibility with models trained on different resolutions.
    - **Characteristics:** Features a diverse set of scene categories and objects in various resolutions.

5. **SVHN (Street View House Numbers)**[168]

    - **Purpose:** Real-world image dataset for digit classification extracted from Google Street View.

- **Characteristics:** Contains over 600,000 digit images that capture a variety of real-world conditions.

6. **Textures**[169]

   - **Purpose:** Utilized for texture recognition and classification tasks.
   - **Characteristics:** Composed of images depicting a wide range of textures and materials.

7. **STL10**[170]

   - **Purpose:** Used for developing unsupervised feature learning, deep learning, and self-taught learning algorithms.
   - **Characteristics:** Consists of 5,000 training images and 8,000 test images in 10 pre-defined classes with a resolution of 96x96 pixels.

8. **Places365**[171]

   - **Purpose:** A comprehensive dataset for scene recognition and context understanding.
   - **Characteristics:** Contains over 1.8 million images from 365 scene categories, offering a large and diverse set of scenes.

9. **iSUN**[172]

   - **Purpose:** A subset of the SUN database, used for assessing model generalization.
   - **Characteristics:** Provides a substantial number of images suitable for robustness and generalization assessments.

The comprehensive Table 3.1 illustrates the diverse array of datasets utilized in assessing the performance of the machine learning model, with CIFAR-10 and CIFAR-100 serving as the in-distribution datasets for training. CIFAR-10 comprises simple, everyday objects with a moderate level of intra-class variability, while CIFAR-100 presents a more challenging scenario with

its finer classification of a hundred categories, demanding the model to discern more subtle differences among classes.

For a robust evaluation, the model is rigorously tested against a suite of 10 Out-of-Distribution (OOD) datasets, each with unique characteristics that challenge the model's generalization capabilities. MNIST and K-MNIST offer a stark contrast to the colored images of CIFAR, presenting monochromatic digit and character images respectively. Fashion-MNIST further extends this challenge with grayscale images of clothing items, testing the model's feature extraction capabilities in a different domain.

LSUN, provided in both cropped and resized variants, introduces complexity with real-world scenes, pushing the model to adapt to broader and more complex spatial arrangements. SVHN and Textures bring real-world variability and pattern recognition into the mix, with the former offering digit images sourced from natural scenes and the latter presenting an array of surface patterns that require the model to learn textural discernment.

STL10 contributes high-resolution images, augmenting the diversity of the visual domain. The inclusion of Places365 offers a plethora of scene categories, broadening the model's exposure to various environmental contexts. iSUN complements the evaluation with additional scene recognition challenges, and the additional use of CIFAR-100 as an OOD set serves to test the model's specificity to the distribution it was trained on.

The model's ability to accurately classify images from CIFAR-10 and CIFAR-100 while effectively identifying and differentiating OOD samples from the 10 test datasets is indicative of its robustness and reliability, key attributes for real-world deployment. This multi-dataset evaluation strategy ensures that the model's performance is not just confined to the data it was trained on but extends to a wide array of unseen, diverse data scenarios.

3.3 Objective I: In-Distribution and Out-Distribution Operational Design Domain

In this section, we outline the methodology for establishing and defining the Operational Design Domain for a machine learning model trained in the recognition of airplane images. Our approach bifurcates into two distinct yet interconnected objectives: the In-Distribution Operational Design Domain and the Out-of-Distribution Operational Design Domain. The former focuses on creating criteria for scenarios closely aligned with the training data, where the model is trained to recognize close-up images of airplanes. This involves specifying a range of opera-

Dataset	Description	# of Images	# of Classes	Resolution
CIFAR-10	Colored natural images	60,000	10	32x32
CIFAR-100	Colored images with fine labels	60,000	100	32x32
MNIST	Grayscale handwritten digits	70,000	10	28x28
K-MNIST	Grayscale images of Kuzushiji characters	70,000	10	28x28
Fashion-MNIST	Grayscale fashion product images	70,000	10	28x28
LSUN (crop)	Cropped images of various scenes	Varied	Varied	Varied
SVHN	Real-world digit images from house numbers	600,000+	10	Varied
Textures	Images of various textures	Varied	Varied	Varied
STL10	Images for unsupervised learning	13,000	10	96x96
Places365	Images of diverse scenes and places	1.8M+	365	Varied
iSUN	Subset of SUN database for scene recognition	Varied	Varied	Varied
LSUN (resize)	Resized images of various scenes	Varied	Varied	Varied

Table 3.1: Overview of Datasets Used for Training and Out-of-Distribution Testing, with CIFAR-10 and CIFAR-100 for In-Distribution Training and 10 Varied Datasets for Comprehensive OOD Evaluation

tional parameters including distances, angles, lighting conditions, and backgrounds, tailored to optimize model accuracy and reliability in these controlled conditions. Conversely, the Out-of-Distribution Operational Design Domain addresses scenarios where the model encounters images significantly different from its training set, such as far-off airplane images. Here, the emphasis is on identifying and characterizing external factors like extreme distances, varied environmental conditions, and unusual perspectives. This duality in approach aims to equip the model with the versatility to maintain performance and robustness, not only in familiar in-distribution contexts but also in challenging out-of-distribution situations.

3.3.1 Methodology

To effectively define the In-Distribution Operational Design Domain and the Out-of-Distribution Operational Design Domain, our approach systematically categorizes and analyzes various scenarios based on three key environmental variables: weather, light, and distance. These variables play a pivotal role in defining the operational conditions and corresponding responses of the machine learning model, especially in the context of recognizing airplane images, as detailed in the accompanying Table 3.2. In particular, for the OOD scenarios, the complexity is intentionally simplified to align with the testing requirements of TinyML and to ensure runtime assurance. This simplification is crucial for evaluating the model's effectiveness in TinyML environments, where computational resources are limited, and for verifying that the model

maintains reliable performance even under less complex, yet essential, out-of-distribution conditions.

Environmental Conditions	
Weather	Rainy
	Cloudy
	Sunny
Light	Day
	Night
Distance	Close
	Far

Table 3.2: Operational Design Domains Based on Scenario Environment, Detailing Variations in Weather Conditions, Lighting Conditions, and Distance Parameters

Operational Design Domain attributes

1. **In-Distribution Operational Design Domain (IDD)**:

    - **Weather Conditions**: For IDD, we focus on scenarios where weather conditions are stable and predictable, such as clear or slightly cloudy weather. This choice is based on the assumption that such conditions are most representative of the training data.
    - **Light Conditions**: The IDD primarily encompasses scenarios with optimal lighting – predominantly daytime settings. The model is trained and optimized for scenarios where visibility is high and shadows or artificial lighting do not significantly affect image recognition.
    - **Distance**: The distance parameter in IDD is set to 'Close', implying that the airplane images the model will recognize are within a certain proximal range. This range is predetermined based on the typical distances at which the model performs with the highest accuracy in controlled environments.

2. **Out-of-Distribution Operational Design Domain (oDD)**:

    - **Weather Conditions**: For oDD, the model is tested against more challenging and less predictable weather scenarios, such as heavy rain or dense cloud cover. These

conditions are selected to assess the model's robustness and reliability in adverse weather situations that deviate from the typical training data.

- **Light Conditions**: In oDD, the focus shifts to include low light or night conditions. This expansion challenges the model to maintain performance when faced with reduced visibility and the presence of artificial lighting, which can introduce new variables and complexities.
- **Distance**: The 'Far' distance setting is critical in oDD. Here, the model encounters airplane images at greater distances than those used in training, pushing its ability to generalize and maintain accuracy when dealing with scaled-down or less-detailed representations of aircraft.

By methodically setting and evaluating these parameters, our approach aims to thoroughly understand and define the operational boundaries for both the In-Distribution and Out-of-Distribution domains. This dual-focused strategy ensures a comprehensive assessment of the model's capabilities and limitations, leading to a more robust and reliable application in real-world scenarios.

3.3.2 Experimental Setup

The primary objective of this experiment is to meticulously curate a dataset comprising 1,438 distinct images of airplanes, each with a resolution of 32x32 pixels. This dataset is specifically crafted for the purpose of training a sophisticated machine learning model. To achieve this, we employ Flight Gear, a highly versatile and realistic flight simulation software, renowned for its accuracy in replicating various aviation scenarios.

The dataset is thoughtfully divided into two principal categories, each representing a unique Operational Design Domain (ODD). The first category, labeled as 'In-Distribution Operational Design Domain' **(label 0)**, includes images that align closely with the typical conditions and scenarios under which the machine learning model is expected to operate. These images are characterized by scenarios that the model will most commonly encounter during its application, ensuring that the model is well-trained for standard operational environments.

Conversely, the second category, known as the 'Out-of-Distribution Operational Design Domain' **(label 1)**, consists of images that depict scenarios substantially different from the model's primary training data. This category is crucial for testing and enhancing the model's ability to generalize and perform accurately in less common or unexpected situations, thereby improving its robustness and reliability.

Each image in both categories is generated through the careful manipulation of various environmental parameters within Flight Gear. These parameters include, but are not limited to, weather conditions (ranging from sunny to cloudy to rainy), time of day (encompassing day and night settings), and the positioning of the airplane relative to the camera (such as varying distances and angles). By systematically altering these variables, we ensure a rich and diverse dataset that thoroughly encompasses both in-distribution and out-of-distribution scenarios.

Simulation Software: FlightGear

- Selection: FlightGear, an open-source flight simulator, is chosen for its versatility and realism in simulating various flight scenarios and environmental conditions.
- Configuration: The software is configured to simulate diverse flight environments, adjusting variables such as weather, lighting, and distance from the camera.

Image Generation Parameters

- Weather Conditions: Images are generated under three weather conditions: rain, cloudy, and sunny.
- Lighting Conditions: Two lighting conditions are considered: day and night.
- Distance: The distance parameter is bifurcated into 'Close' for in-distribution images and 'Far' for out-of-distribution images.

In-Distribution Operational Design Domain (Label 0)

- Criteria: Images are captured with the airplane in close proximity under varying weather and lighting conditions. These scenarios closely resemble the training data conditions.
- Image Capture: A total of **(700+)** images are generated, ensuring a balanced representation of each combination of weather and lighting conditions at close distances.

Out-of-Distribution Operational Design Domain (Label 1)

- Criteria: In this domain, images are captured with the airplane at a far distance, under different weather and lighting conditions, representing scenarios significantly different from the training data.
- Image Capture: A total of **(700+)** images are generated to capture the diversity of out-of-distribution scenarios, emphasizing far distances in various environmental conditions.

Data Collection Process

- Automated Capture: An automated script is used to capture images from FlightGear, ensuring consistent image quality and scenario representation.
- Image Labeling: Each image is manually labeled as '0' for in-distribution or '1' for out-of-distribution, based on the predefined criteria.

3.3.3 Results

The results of our objective into the Operational Design Domain for a machine learning model trained in airplane image recognition offer insightful findings in two distinct set of images but interrelated areas: the In-Distribution Operational Design Domain and the Out-of-Distribution Operational Design Domain. In the In-Distribution domain, we focused on establishing criteria for scenarios where the model recognizes close-up images of airplanes, optimizing it for a range of distances, angles, lighting conditions, and backgrounds. This optimization was aimed at ensuring high accuracy and reliability for scenarios that closely align with the training data of the model, as shown in Figure 3.4. In contrast, for the Out-of-Distribution domain, we developed requirements to effectively handle scenarios where the model encounters far-off images of airplanes, differing significantly from its training data as shown in Figure 3.5 . This involved a meticulous identification and characterization of external factors such as extreme distances, varied environmental conditions, and atypical perspectives. The goal was to train the model to maintain its performance and robustness, even when processing data that falls outside its primary training distribution. The results from both domains provide a comprehensive overview of the model's capabilities and limitations in varied operational scenarios,which will be later utilized to train and evaluate the model's accuracy for ensuring runtime performance in **Objective III** as described in Section 3.5.

Figure 3.4: A selection of images from the In-Distribution Operational Design Domain dataset, depicting airplanes under varied Weather Conditions (Rain, Cloudy, Sunny), Lighting Conditions (Day, Night), and captured at Close Distances, demonstrating the diversity and complexity of scenarios for machine learning model training.

Figure 3.5: A Selection of Images from the Out-of-Distribution Operational Design Domain Dataset, Demonstrating Airplanes in Far-Off Scenarios Under Diverse Weather Conditions (Rain, Cloudy, Sunny), Various Lighting Conditions (Day, Night), and Emphasizing Distance Parameters (Far)

3.4 Objective II: The Multi-Layer Framework- Fast TinyML OOD Detector (FTO)

In our endeavor to improve the detection of Out-of-Distribution (OOD) in machine learning, specifically for TinyML applications, we introduce a groundbreaking framework: the Multi-Layer Framework-Fast TinyML OOD Detector (FTO). Central to our proposed solution is the integration of a low-complexity detector after each layer of the Deep Neural Network (DNN), designed to facilitate an early exit during inference when encountering an OOD input sample. This detector, leveraging internal representations, employs a logistic regression classifier to discern between in-distribution and out-of-distribution inputs. A key feature of our approach is its plug-and-play nature, which allows for seamless integration without necessitating modifications to existing off-the-shelf models.

The design of our DNN classifier, as described in Section 3.4.1, is specifically customized to improve the precision of out-of-distribution detection. This method utilizes data from the preceding layer for early inference termination, a crucial factor for applications requiring fast inference times. Our method stands out by efficiently integrating into the DNN structure

without the need for additional processing steps. Moving beyond the typical softmax output used in conventional DNN classifiers, which has shown limited effectiveness in OOD detection, our study focuses on internal DNN representations, particularly the penultimate layer. This change is supported by previous research that has analyzed these representations to understand the geometric nature of the internal structures of DNN.

Innovatively, our architecture replaces the conventional final fully connected layer with a Gaussian layer. This approach is based on previous studies but with a significant simplification for practical application: instead of training a DNN from scratch, we fine-tune an existing trained model. We replace the last layer with a Gaussian layer and manually initialize the layer parameters using statistical estimations derived from the penultimate representation of the training data. This strategy not only enhances the model's OOD detection capabilities, but also aligns with the pragmatic requirements of real-world TinyML applications.

3.4.1 Model

The proposed framework highlights a novel approach to integrating Out-of-Distribution (OOD) detection within a Deep Neural Network (DNN) using early exit strategies and an efficient detection mechanism. This adaptive inference network is equipped with (k) OOD detectors, each placed at specific depths within the network architecture, as illustrated in the lower section of the framework Figure 3.6. The process begins with the input being fed into the network, where a dynamic complexity score is calculated to determine the appropriate exit point during inference. This score helps in assessing whether the input can be reliably classified at an early stage or if it needs deeper analysis for accurate classification.

At each potential exit point, the network employs an OOD detector. These detectors are designed to leverage the information from both the current and the preceding layers. By analyzing the cumulative insights gained across these layers, the detectors can effectively distinguish between in-distribution data (data similar to what the network was trained on) and OOD data (novel or unexpected inputs).

In the final stage of the network, the typical architecture comprises a linear transformation, often implemented as a fully connected layer, followed by a softmax activation function. This conventional setup is retrained to ensure that the network can still perform standard classification tasks effectively, in addition to its enhanced capability of OOD detection.

This integration of OOD detection within the DNN framework aims to improve the network's robustness and reliability, particularly in scenarios where encountering unexpected data

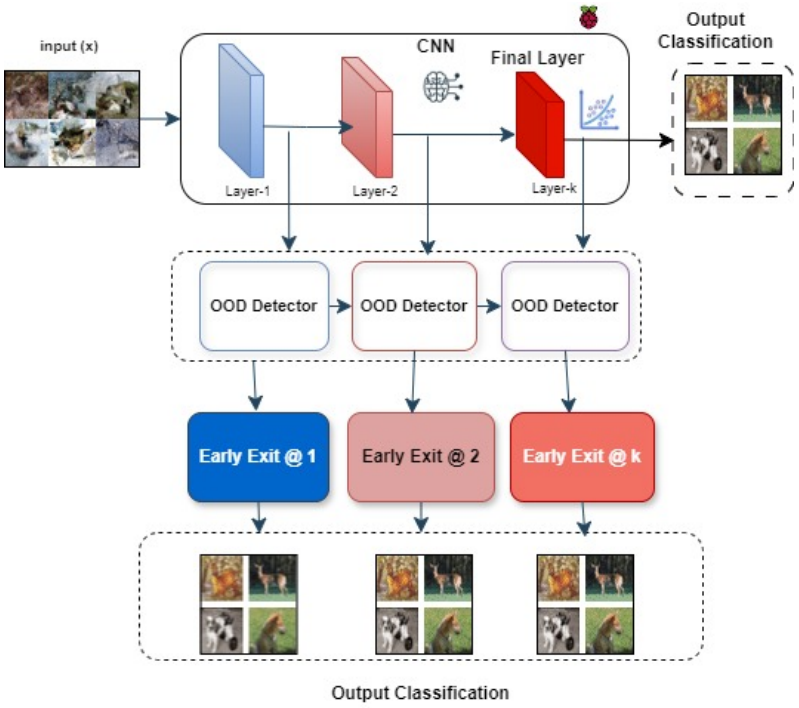

Figure 3.6: The framework overview showcases the use of early exit strategies and efficient detection method that seamlessly integrates OOD detection within the DNN. The adaptive inference network incorporates k OOD detectors, strategically positioned at distinct depths within the network (depicted at the bottom). Given an input, a dynamic complexity score is employed to decide the exit point during the inference process. At each exit, an OOD detector integrates insights from both the current and previous layers to distinguish between in-distribution and OOD data.Replacing the traditional final layer with a Gaussian layer for enhanced detection capabilities.

inputs is likely. By enabling early exits based on the complexity score, the network optimizes its computational efficiency, only engaging its deeper layers when necessary. This approach not only conserves computational resources but also potentially accelerates the inference process, making the network more suitable for real-time applications.

In the realm of Deep Neural Network (DNN) classifiers, traditional architectures culminate in a linear transformation followed by softmax activation. However, the softmax output's efficacy in Out-of-Distribution (OOD) detection is limited. Building on this, our proposed framework pivots from softmax to an emphasis on the DNN's internal representations, particularly the penultimate layer.

Enhanced Gaussian Layer Architecture

Our architecture integrates a Gaussian layer at the end, diverging from the standard softmax approach. The mean (μ) and covariance (Σ) parameters for each class c are computed as follows:

- Mean calculation for class c:

$$\mu_c = \frac{1}{N_c} \sum_{x \in S_c} h(x) \tag{3.1}$$

- Covariance calculation for class c:

$$\Sigma_c = \frac{1}{N_c} \sum_{x \in S_c} (\mu_c - h(x))(\mu_c - h(x))^T \tag{3.2}$$

Here, S_c is the set of samples belonging to class c, and $h(x)$ represents the penultimate representation of x.

Log Likelihood Ratio (LLR) Test

We utilize a Log Likelihood Ratio (LLR) test, based on the Gaussian log likelihoods: - Predicted class log likelihood:

$$L_{pred}(x) = \max_c \log \mathcal{N}(h(x); \mu_c, \Sigma_c) \tag{3.3}$$

- Other classes' average log likelihood:

$$L_{other}(x) = \frac{1}{C-1} \sum_{c' \neq c} \log \mathcal{N}(h(x); \mu_{c'}, \Sigma_{c'}) \tag{3.4}$$

The LLR score is defined as the difference between these two metrics:

$$LLR(x) = L_{pred}(x) - L_{other}(x) \tag{3.5}$$

This LLR score, serving as an efficient OOD detection metric, distinguishes in-distribution samples (higher LLR) from OOD samples (lower LLR).

Early Exit Strategy

Our framework introduces an early exit strategy for "easy" OOD instances at initial layers, allowing deeper layers to handle more complex cases. Each layer has a binary classifier:

- Early exit detector at layer I:

$$E_i(x; \alpha_i) = \begin{cases} \text{in, if } T_i(x; \alpha_i) \geq \delta_i \\ \text{out, if } T_i(x; \alpha_i) < \delta_i \end{cases} \tag{3.6}$$

E_i represents the early exit detector at the ith layer.
x is still the input to the function.
α_i symbolizes the new parameters for the ith layer's function.
T_i is the new notation for the scoring function at the ith exit.
δ_i is the new symbol for the threshold at layer i.

Feature Extraction and Logistic Regression Classifier

The feature extraction at layer l is defined as the maximum logarithmic likelihood for the membership of the class:

- Feature extraction at layer l:

$$F(x, l) = \max_c \log \mathcal{N}(h[l](x); \mu[l]_c, \Sigma[l]_c) \tag{3.7}$$

The score function for the logistic regression classifier combines these features:

- Logistic regression score function:

$$\text{score}(x) = \sigma\left(\sum_{l=1}^{L} w_l \cdot F(x, l) + w_{L+1} \cdot LLR(x)\right) \tag{3.8}$$

3.4.2 Algorithm I:Fast TinyML OOD Detector

The Fast TinyML Out-of-Distribution Detection (FTO) algorithm 3.1 presents a sophisticated approach for identifying OOD samples in neural networks, particularly tailored for TinyML applications where computational efficiency is paramount. This algorithm operates on a multi-layer structure, enhancing the detection process with several strategic components.

At the outset, each input sample is processed starting from the first convolutional layer of the Deep Neural Network (DNN). The algorithm comes with a set number of out-of-distribution detectors, carefully placed at designated layers in the deep neural network. As the input passes through these layers, the algorithm calculates the second-to-last representation of the input at each level, which is a critical process laying the groundwork for the following out-of-distribution detection.

A pivotal aspect of the FTO algorithm is its early exit mechanism. This mechanism is activated at each layer, where the algorithm evaluates the need for further processing. If the input is determined to be OOD at any given layer, as indicated by the corresponding OOD detector, the algorithm immediately classifies it as such and halts further processing. This early exit strategy is particularly beneficial as it prevents unnecessary computations, especially for inputs that can be readily identified as OOD in the initial layers.

Nevertheless, in case the input is not identified as OOD in the initial layers, the algorithm proceeds to move through the DNN. It carefully interacts with the OOD detector in each layer, utilizing the knowledge gathered from both the current and previous layers. This stratified method allows for a more detailed and comprehensive analysis of the input, thereby improving the precision of OOD detection.

The process culminates at the final Gaussian layer of the DNN. Here, the algorithm combines the results from the Log Likelihood Ratio (LLR) statistical test and an additional OOD neuron. This combination provides a robust mechanism for final OOD determination, ensuring that even the most complex OOD instances are accurately detected.

3.4.3 Experimental Setup

In our experimental setup, we rigorously assess the Fast TinyML Out-of-Distribution Detection (FTO) method by comparing it against a suite of state-of-the-art OOD detection techniques explained in Section2.2.10 including Maximum Softmax Probability (MSP), Multi-layer OOD Detection, ODIN, Channel-Attention-based Neural Mean Discrepancy (CD-NMD), Out-of-Distribution Discernment Layer, Mahalanobis Distance, Energy-based OOD Detec-

Algorithm 3.1 Fast TinyML Out-of-Distribution Detection (FTO)

Require: Input sample x, Deep Neural Network (DNN) with k OOD detectors
Ensure: Classification of x as in-distribution or OOD

Initialization:
Load the DNN with k OOD detectors at designated layers.
Set parameters for each layer: $\mu_c, \Sigma_c, \gamma_i$ based on training data.
For each input sample x:
Start at the first convolutional layer of the DNN.
Compute the penultimate representation $h(x)$ at each layer.
Early Exit Detection:
for each layer i (starting from the first)
    Calculate the feature extraction $F(x, i)$ using Equation 3.7.
    Evaluate the early exit condition using Equation 3.6:
    if $G_i(x; \theta_i) = \text{out}$
        Classify x as OOD and exit.
    else
        Continue to the next layer.
    end if
end for
Final Layer Processing:
if no early exit occurs
    Compute the mean μ_c and covariance Σ_c for the final layer using Equations 3.1 and 3.2.
    Calculate the predicted class log likelihood $L_{pred}(x)$ using Equation 3.3.
    Compute the average log likelihood for other classes $L_{other}(x)$ using Equation 3.4.
    Determine the LLR score using Equation 3.5.
end if
OOD Decision Making:
Apply the logistic regression score function score(x) using Equation 3.8.
if score(x) indicates OOD
    Classify x as OOD.
else
    Classify x as in-distribution.
end if
Output:
for each input sample
    Output whether it is in-distribution or OOD.
end for

tion, and Softmax Response all these methods use the same four-layer convolutional neural network (CNN) as the backbone. The CIFAR-10 and CIFAR-100 datasets form the basis of our in-distribution (ID) training, while a comprehensive evaluation is conducted across 10 distinct OOD test datasets—MNIST, K-MNIST, Fashion-MNIST, LSUN (crop and resize), SVHN, Textures, STL10, Places365, and iSUN.

3.4.4 Architecture & Training

For consistency, all images are resized to a uniform 32x32 resolution. This standardization is crucial given the varying distributions of the selected Out-of-Distribution (OOD) datasets. CIFAR-100 closely mirrors the distribution of CIFAR-10, while the SVHN dataset deviates most distinctly from CIFAR-10's data distribution. To thoroughly assess the robustness of OOD detection, we also perform evaluations on a mixed set of these datasets, thereby simulating a more complex and heterogeneous OOD environment.

In terms of hardware, we opted for the Arduino Nano 33 BLE Sense [173] for our experimental setup. This choice is significant as it imposes a constraint of 1MB of flash memory and 256KB of SRAM, simulating a realistic and challenging scenario for TinyML applications. There is no reliance on any external storage such as an SD card. Instead, the Arduino is interfaced with a laptop from which it receives images one at a time through serial communication. After processing each image, the Arduino sends a request back to the laptop to receive the next one as shown in Figure 3.7. This interaction underscores the real-time processing capability of the system. Moreover, we account for the limited computational capability of microcontrollers (MCUs) by pre-calculating the complexity of each image and sending this information alongside the image data to the Arduino. TThe training procedures are thorough and uniform. The model undergoes training with the in-distribution (ID) dataset pairs and is kept unchanged thereafter, guaranteeing that the evaluation of the FTO detectors is carried out on a steady and unaltered model. Specifically, we train the FTO detectors on 30,000 images from both the ID and OOD datasets and subsequently test them on an independent set of 5,000 images from each category. The training process spans 100 epochs, managed by the Adam [174] optimization algorithm with batches of 32 images each, striking a balance between computational efficiency and learning performance. This methodical approach to training and evaluation is designed to rigorously test the capabilities of the FTO method in distinguishing between in-distribution and out-of-distribution samples under resource-constrained conditions.

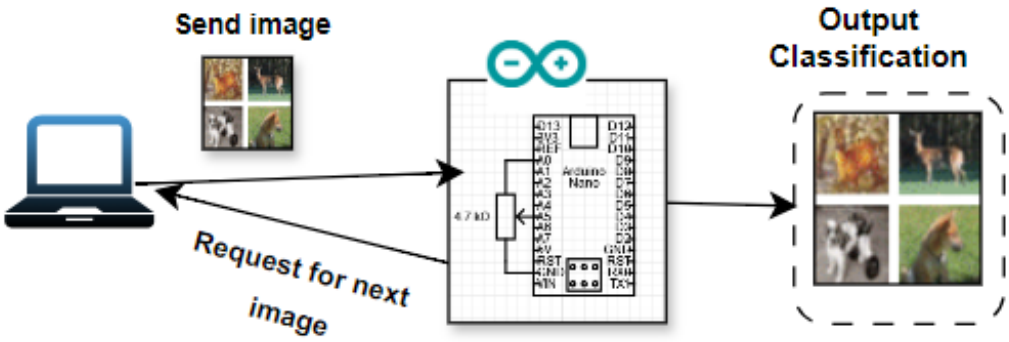

Figure 3.7: The Figure illustrates the workflow between the Arduino and a laptop for realtime image processing and data exchange. This process involves the Arduino receiving images from the laptop via serial communication, processing each image, and then requesting the next image

3.4.5 Evaluation Methodology

In the evaluation of the Fast TinyML Out-of-Distribution Detection (FTO) method and other baseline methods, we utilize a comprehensive set of metrics:

1. **Number of Computational FLOPS During Inference Time:** The Floating Point Operations Per Second (FLOPS) measures the computer's performance capability in handling floating-point calculations. In our context, we specifically assess the number of computational FLOPS required during the model's inference phase. This metric is crucial in TinyML environments where computational resources are inherently constrained. An optimal model is characterized by a lower FLOPS requirement, signifying efficient processing suitable for real-time applications and devices with limited processing capabilities.

$$\text{FLOPS} = \frac{\text{Number of Floating Point Operations (FLOPs)}}{\text{Time (seconds)}} \tag{3.9}$$

2. **False Positive Rate (FPR95) on OOD Data:** The FPR95 refers to the False Positive Rate when the True Positive Rate (TPR) for in-distribution data is fixed at 95%. Mathematically, it is represented as:

$$FPR95 = \mathbb{P}(\text{False Positive} | \text{True Positive Rate} = 0.95) \tag{3.10}$$

This metric evaluates the model's specificity, or its ability to accurately reject OOD instances without misclassifying them as in-distribution. Lower values of FPR95 are

preferable, indicating higher model accuracy for ID samples and effective OOD detection.

3. **Area Under the Receiver Operating Characteristic Curve (AUROC):** The AUROC is a widely recognized performance metric for binary classification at various threshold settings. It is calculated as the area under the ROC curve, which plots the True Positive Rate against the False Positive Rate at different thresholds:

$$AUROC = \int_0^1 \text{TPR}(t)\, dt \tag{3.11}$$

where t represents the threshold. An AUROC value of 1 indicates perfect model performance, while a value of 0.5 suggests no discriminative power. A high AUROC value in OOD detection implies that the model can effectively discriminate between in-distribution and out-of-distribution data across all thresholds.

3.4.6 Results

The results in Table 3.3, Figure 3.8,Figure 3.9 and Figure 3.10 showcases the comparative analysis of Out-of-Distribution (OOD) detection methods across CIFAR-10 and CIFAR-100 datasets reveals that our FTO method exhibits superior performance. On the CIFAR-10 benchmark, FTO stands out with the highest AUROC score of **0.9126**, indicating its robust ability to distinguish between in-distribution and out-of-distribution data. Furthermore, FTO achieves the highest in-distribution accuracy (ID Acc) at an impressive **94.13%**. This is complemented by its computational efficiency, requiring the lowest number of FLOPs at **0.793** $\times 10^8$, which is a significant reduction compared to other evaluated methods.

In the context of CIFAR-100 dataset, the FTO method similarly excels in terms of AUROC, reaching **0.8947**, which is significantly higher than the next closest competitor, ODIN, at **0.848**. It also shows an impressive ID Acc of **85.76%**, outpacing the majority of methods. Noteworthy is the fact that FTO achieves this while maintaining a low computational footprint, reflected in the reduced FLOPs.

Collectively, these results highlight the FTO method's efficacy and efficiency in OOD detection, endorsing its application in scenarios where computational resources are constrained, and accurate, timely detection of OOD samples is critical. The method's adaptability is further underscored by its consistent performance across a range of OOD test datasets, includ-

In-distribution (ID)	Method	FLOPs $\times 10^8 \downarrow$	AUROC$\uparrow$	ID Acc $\uparrow$
CIFAR-10	MSP	1.051	0.8972	65.37
	ODIN	0.884	0.9033	71.24
	Mahalanobis	0.878	0.8284	74.26
	Energy	0.897	0.9048	91.09
	MOOD	0.869	0.8471	75.10
	CA-NMD	0.884	0.8531	75.43
	OODL	0.984	0.8451	75.26
	FTO (ours)	**0.793**	**0.9126**	**94.13**
CIFAR-100	MSP	1.051	0.7833	68.43
	ODIN	0.98	0.848	73.43
	Mahalanobis	0.880	0.7380	85.34
	Energy	1.050	0.8451	75.43
	MOOD	0.794	0.8400	73.84
	CA-NMD	0.774	0.8507	72.99
	OODL	0.788	0.8298	75.26
	FTO (ours)	**0.689**	**0.8947**	**85.76**

Table 3.3: OOD Detection Performance Comparison. This table presents a comprehensive evaluation of various Out-of-Distribution (OOD) detection methods, quantified by metrics such as FLOPs during inference, AUROC scores, and the accuracy of identifying in-distribution data (ID Acc). Methods that have achieved the highest performance in each metric are emboldened, indicating their superiority in the respective category. The comparison serves to highlight the efficiency and effectiveness of the detection mechanisms within the context of OOD identification tasks.

ing MNIST, K-MNIST, Fashion-MNIST, LSUN (crop and resize), SVHN, Textures, STL10, Places365, CIFAR-10, and iSUN.

Overall, the results indicates that FTO (ours) provides a balance of efficiency and effectiveness, outperforming other methods in key areas, especially in computational efficiency and the capability to detect out-of-distribution data as measured by AUROC. ID accuracy remains high across both datasets, though there is more competition among methods on CIFAR-100.

3.5 Objective III:Multi-Layer Early Exit for OOD Detection with LOF in DNNs (MELOD)

In this section, we venture into the nuanced realm of integrating a multi-layer early exit strategy within Deep Neural Networks (DNNs) for the explicit purpose of enhancing Out-of-Distribution (OOD) detection. This approach resonates deeply with the ethos of TinyML, prioritizing com-

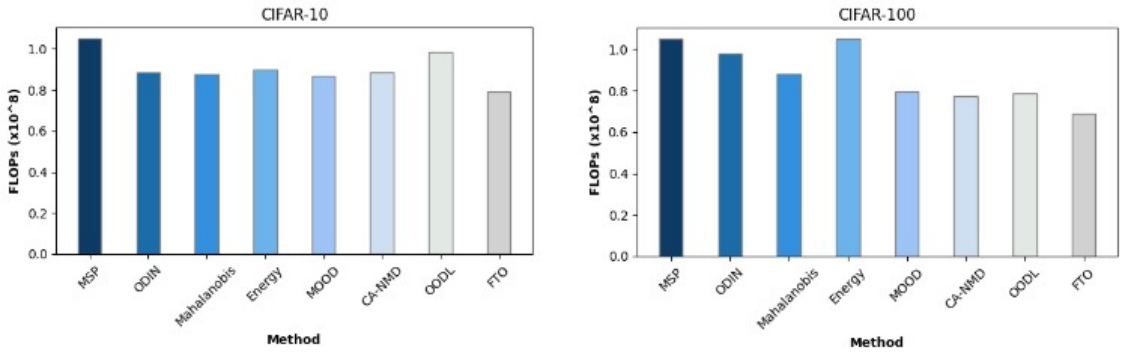

Figure 3.8: The Figure illustrates the performance metric of various OOD detection methods on CIFAR-10 and CIFAR-100. The subplots show the number of **FLOPs** $\times 10^8$ among the methods analyzed, represented in distinct light colors for visual clarity. Lower FLOP values indicate better performance, with the best performing methods denoted by their respective color

.

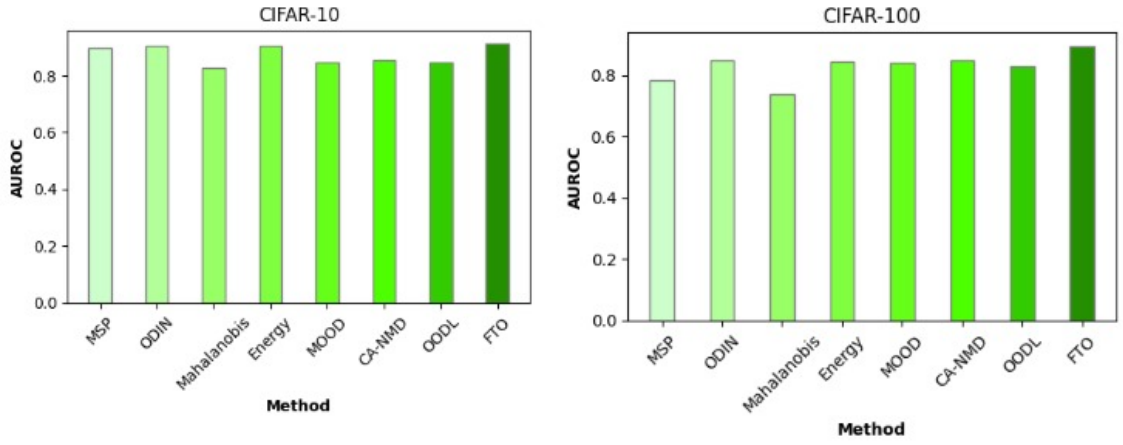

Figure 3.9: The Figure illustrates the performance metric of various OOD detection methods on CIFAR-10 and CIFAR-100. The subplots show the **Area Under the Receiver Operating Characteristic Curve (AUROC)** among the methods analyzed, represented in distinct light colors for visual clarity. Higher AUROC values indicate better performance, with the best performing methods denoted by their respective colors

.

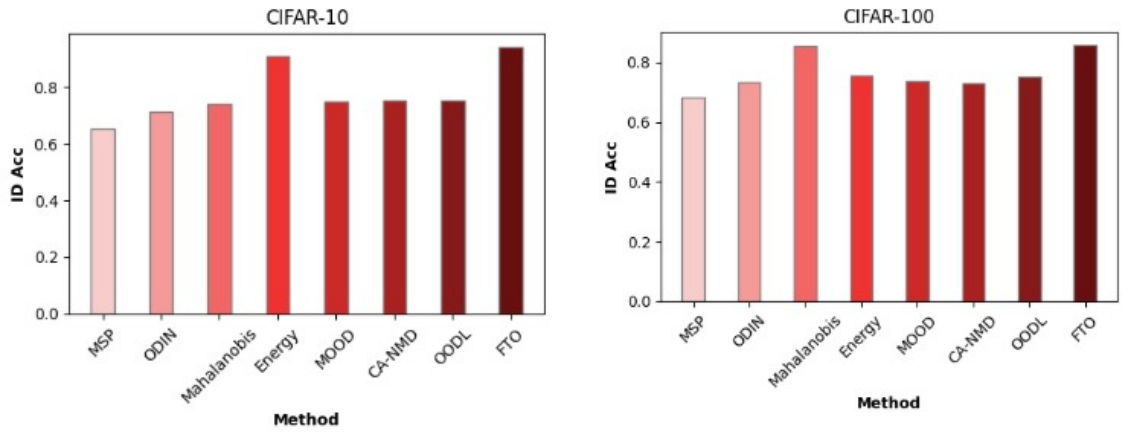

Figure 3.10: The Figure illustrates the performance metric of various OOD detection methods on CIFAR-10 and CIFAR-100. The subplots show the **In-Distribution Accuracy (ID Acc)** among the methods analyzed, represented in distinct light colors for visual clarity. Higher ID Acc values indicate better performance, with the best performing methods denoted by their respective colors

.

putational efficiency in environments constrained by limited resources. Our exploration will cover the intricacies of incorporating outlier detection mechanisms across multiple hidden layers of a DNN, employing a runtime monitor rooted in the Local Outlier Factor (LOF). This pivotal technique serves to pinpoint and manage data anomalies that stray from the expected norm. We will elaborate on how this strategic method significantly expedites the OOD detection process, consequently alleviating the computational load often entailed by deep neural processing. This aspect holds particular significance in fields like avionics systems, where the swift identification of irregular data is crucial to ensuring the steadfast reliability and safety of machine learning applications. Through the implementation of this method, we aim to showcase the potency of early exits in DNNs as a means to discern OOD data, thereby bolstering the model's agility and resilience, even within the most demanding of operational contexts.

3.5.1 Model

In TinyML applications, we implement the model presented in section 3.3.1, integrating the Local Outlier Factor (LOF) for runtime Out-of-Distribution (OOD) detection. The LOF is applied within a DNN, which is mathematically represented as

$$\mathbf{Y} = G(\mathbf{X}; \Theta) = g^{(n)} \circ g^{(n-1)} \circ \ldots \circ g^{(1)}(\mathbf{X}) \tag{3.12}$$

where $g^{(n)}$ denotes the n-th layer of DNN G; W_n, B_n denote the weight matrix and bias vector of the n-th layer, respectively; $\mathbf{X}_{n-1}$ denotes the vector of activation values output from the $(n-1)$-th layer; $\sigma^{(n)}()$ denotes the nonlinear activation function at the n-th layer, e.g., ReLU. For simplicity, we adopt a uniform notation for both convolutional and fully-connected layers. The output of the n-th layer for activations from the $(n-1)$-th layer is given by:

$$g^{(n)}(\mathbf{X}_{n-1}) = \sigma^{(n)}(W_n \cdot \mathbf{X}_{n-1} + B_n) \tag{3.13}$$

where $g^{(n)}$ represents the function of the n-th layer in the DNN. Here, W_n is the weight matrix and B_n is the bias vector associated with the n-th layer. The vector $\mathbf{X}_{n-1}$ contains the activation values output by the $(n-1)$-th layer. The function $\sigma^{(n)}()$ denotes the nonlinear activation function applied at the n-th layer, such as ReLU (Rectified Linear Unit). This notation is consistent across both convolutional and fully-connected layers, with the understanding that convolutional operations can also be expressed via matrix multiplication.

The novel framework as shown in Figure 3.11 presents a model for detecting Out-of-Distribution (OOD) data points within a given dataset using a Deep Neural Network (DNN). The model initiates with a dataset X consisting of data points $x_1, x_2, \ldots, x_N$. Central to this model is a DNN, denoted as G, composed of multiple layers $g^{(1)}, g^{(2)}, \ldots, g^{(n)}$. Each layer is defined by its unique weight matrices $W_1, W_2, \ldots, W_n$, bias vectors $B_1, B_2, \ldots, B_n$, and nonlinear activation functions $\sigma^{(1)}, \sigma^{(2)}, \ldots, \sigma^{(n)}$.

In this framework, each data point x_p in X is processed through the DNN. This process involves sequentially transforming the data point using the weight matrix and bias vector of each layer, followed by the application of a nonlinear activation function. Post-processing, the model calculates the k-distance neighborhood of the output from the DNN's final layer for each data point. Utilizing this neighborhood, the model ascertains the reachability distance, local reachability density (LRD), and then the Local Outlier Factor (LOF) score for every data point.

A significantly high LOF score for a data point signifies its outlier status, indicating a substantial deviation from the dataset's distribution. In contrast, a data point with a non-significant LOF score is considered in-distribution (ID), aligning with the expected dataset distribution.

Furthermore, the model incorporates OOD detectors in each layer of the DNN, applied in reverse order starting from the output layer. This structure improves the efficiency of OOD detection. Detection of OOD at any layer leads to the early termination of further analysis, optimizing the detection process.

The framework integrates OOD detection into the DNN architecture, enhancing its robustness and reliability, especially crucial for handling unexpected data inputs. The early exit strategy, based on complexity scores, boosts computational efficiency by engaging deeper layers only when necessary, conserving resources, and potentially hastening inference, making it suitable for real-time applications. This framework marks a shift from reliance on softmax outputs for OOD detection to focusing on the DNN's internal representations, with particular emphasis on the penultimate layer, facilitating efficient and effective OOD detection within the constraints of TinyML environments.

Local Outlier Factor (LOF)

Local Outlier Factor (LOF) [175] is a method used to identify anomalies by measuring the local deviation of density of a data point with respect to its neighbors. An anomaly is characterized by having a significantly lower density than its neighboring points. The number of neighbors, denoted by the parameter m , is a crucial hyperparameter in the algorithm. The following steps outline the LOF algorithm:

1. m-Distance: For any data point x_p in the set X, where $p = 1, 2, \ldots, N$, the m-distance, $d_m(x_p)$, is defined as:

$$d_m(x_p) = \min \left\{ d \mid \text{at least } m \text{ points } x_q \in X \setminus \{x_p\} \text{ satisfy } d(x_p, x_q) \leq d \right\}. \quad (3.14)$$

2. m-Distance Neighborhood: The m-distance neighborhood of x_p, $N_m(x_p)$, is:

$$N_m(x_p) = \{x_q \in X \setminus \{x_p\} \mid d(x_p, x_q) \leq d_m(x_p)\}. \quad (3.15)$$

3. Reachability Distance: The reachability distance between x_p and x_q is:

$$rd_m(x_p, x_q) = \max\{d_m(x_q), d(x_p, x_q)\}. \quad (3.16)$$

4. Local Reachability Density (LRD): The LRD of x_p is the inverse of the average reachability distance of its m-distance neighborhood:

$$lrd_m(x_p) = \left(\frac{1}{|N_m(x_p)|} \sum_{x_q \in N_m(x_p)} rd_m(x_p, x_q) \right)^{-1}. \quad (3.17)$$

5. Local Outlier Factor (LOF): The LOF score for x_p, $LOF_m(x_p)$, is the ratio of the average

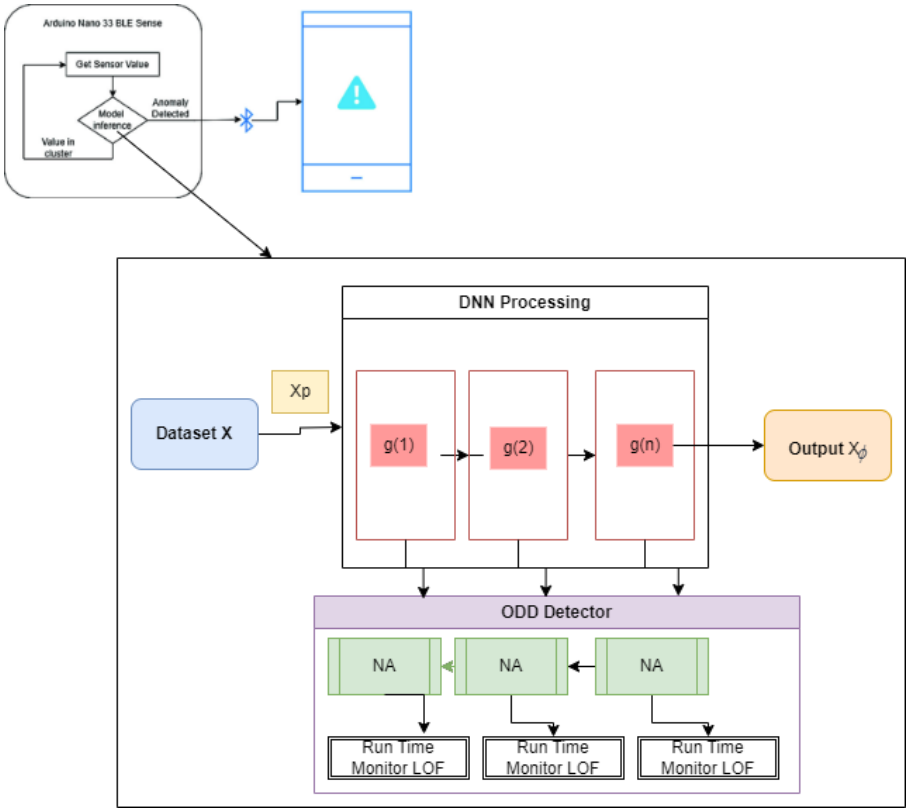

Figure 3.11: This diagram encapsulates the workflow of an algorithm designed for Out-of-Distribution (OOD) detection within a dataset, utilizing a Deep Neural Network (DNN) named G. The process begins with a dataset X consisting of data points $x_1, x_2, \ldots, x_N$, which are sequentially processed through the DNN's multiple layers ($g^{(1)}$ to $g^{(n)}$), each equipped with specific weight matrices (W_1 to W_n) and bias vectors (B_1 to B_n), and governed by nonlinear activation functions ($\sigma^{(1)}$ to $\sigma^{(n)}$). The model assesses each data point's neighborhood (NA) using a predetermined parameter m, calculates the reachability distances, local reachability density (LRD), and the Local Outlier Factor (LOF) score to discern outliers from in-distribution data. OOD detectors are strategically placed within each layer for enhanced detection accuracy, operating in reverse order to ensure computational efficiency. The diagram delineates this intricate process from the initial data input, through DNN processing, neighborhood analysis, LOF score computation, to the final determination of each data point's OOD status, thereby illustrating the comprehensive mechanism of the algorithm in detecting outliers within the computational confines of TinyML environments.

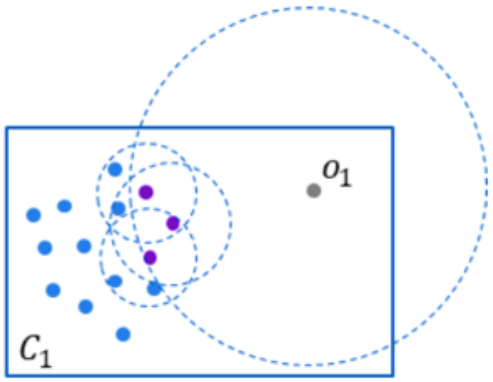

Figure 3.12: Figure illustrates the dataset, utilizing a parameter $m = 3$ for neighborhood analysis. Each circle, centered on a data point x_p, visualizes the m-distance neighborhood, highlighting the spatial relationships within the dataset. Notably, the outlier o_1 is characterized by a substantial m-distance, indicating a pronounced separation from its three nearest neighbors (x_q), depicted in purple, within an expansive circle. To evaluate o_1's LOF score ($LOF_m(o_1)$), we initially calculate the reachability distances ($rd_m(o_1, x_q)$) from o_1 to its neighbors, revealing significantly large values. Subsequently, the local reachability density ($lrd_m(o_1)$) of o_1 is determined, revealing a markedly low density. In contrast, each of o_1's three neighbors undergoes a similar assessment for lrd_m, each yielding considerably higher densities. Ultimately, by calculating o_1's LOF score ($LOF_m(o_1)$) and finding it substantially elevated, o_1 is conclusively identified as an outlier.

LRD of its m-nearest neighbors to its own LRD:

$$LOF_m(x_p) = \frac{\sum_{x_q \in N_m(x_p)} lrd_m(x_q)}{|N_m(x_p)| \cdot lrd_m(x_p)}. \tag{3.18}$$

A high LOF score indicates that x_p has a lower density compared to its neighbors, marking it as a potential outlier as shown in Figure 3.12.

3.5.2 Algorithm II:Multi-Layer Early Exit for OOD Detection with LOF in DNNs (MELOD)

The algorithm 3.2 describes model for detecting Out-of-Distribution (OOD) data points within a given dataset using a Deep Neural Network (DNN). The model begins with a dataset X comprising a set of data points $x_1, x_2, \ldots, x_N$. The core of this model is a DNN, denoted as G, which consists of multiple layers $g^{(1)}, g^{(2)}, \ldots, g^{(n)}$. Each of these layers is characterized by its own weight matrices $W_1, W_2, \ldots, W_n$ and bias vectors $B_1, B_2, \ldots, B_n$, along with nonlinear activation functions $\sigma^{(1)}, \sigma^{(2)}, \ldots, \sigma^{(n)}$.

For each data point x_p in the dataset X, the model processes it through the DNN. This involves passing the data point through each layer of the network, where it is transformed by the

respective weight matrix and bias vector, and then passed through a nonlinear activation function. After processing through the DNN, the model computes the k-distance neighborhood of the output from the last layer of the DNN for each data point. Based on this neighborhood, the model calculates the reachability distance, local reachability density (LRD), and subsequently the Local Outlier Factor (LOF) score for each data point.

The LOF score is crucial as it helps in determining whether a data point is an outlier. A significantly high LOF score indicates that the data point is an outlier, that is, it deviates notably from the distribution of the dataset. Conversely, if the LOF score is not significantly high, the data point is considered to be in-distribution (ID), meaning it conforms to the expected distribution of the dataset.

Additionally, the model includes Out-of-Distribution detectors for each layer of the DNN. These detectors are applied in reverse order starting from the output layer, which enhances the efficiency of detecting OOD instances. If any of these detectors flag a data point as OOD, the analysis for that point is terminated early, thus optimizing the process. distributions.

3.5.3 Experimental Setup

In our experimental setup, we meticulously evaluate the Multi-Layer Early Exit for OOD Detection with LOF in DNNs method by comparing it to a range of state-of-the-art techniques for OOD detection as detailed in Section 2.2.10. These techniques include Maximum Softmax Probability (MSP), Multi-layer OOD Detection, ODIN, Mahalanobis Distance, Energy-based OOD Detection, and Softmax Response. All these methods are implemented using the same four-layer convolutional neural network (CNN) as the backbone.

In our research framework, we leverage the datasets defined in Section 3.2 for both assessments of In-Distribution Operational Design Domain (ID) and Out-of-Distribution Operational Design Domain (OOD). Specifically, the datasets designated in Section 3.3 for The foundation of our in-distribution (ID) training is the In-Distribution Operational Design Domain.This approach ensures that the model is thoroughly trained on data samples that are representative of the expected operational environment and scenarios.

Similarly, the datasets identified in Section 3.3 for Out-of-Distribution Operational Design Domain are utilized to evaluate the model's performance in recognizing and handling OOD instances. This comprehensive testing against diverse OOD datasets is crucial to validate the model's robustness and reliability, particularly in scenarios where it may encounter data that significantly deviates from the patterns and characteristics seen during the training phase.

Algorithm 3.2 Multi-Layer Early Exit for OOD Detection with LOF in DNNs

Given:
A dataset X containing data points $x_1, x_2, \ldots, x_N$
A Deep Neural Network (DNN) G with layers $g^{(1)}, g^{(2)}, \ldots, g^{(n)}$
Weight matrices $W_1, W_2, \ldots, W_n$ and bias vectors $B_1, B_2, \ldots, B_n$
Nonlinear activation functions $\sigma^{(1)}, \sigma^{(2)}, \ldots, \sigma^{(n)}$ for each layer
A parameter m for the number of neighbors in the hood of each data point
Output:
Out-of-Distribution (OOD) detection for each data point in the dataset
Procedure:
for each input sample x_p in X
    Process x_p through the DNN:
    $X_0 \leftarrow x_p$
    for $\varphi = 1$ to n
        $X_\varphi \leftarrow \sigma^{(\varphi)}(W_\varphi \cdot X_{\varphi-1} + B_\varphi)$
    end for
    Compute the k-distance neighborhood of X_n, $N_k(X_n)$
    for each point x_q in $N_m(X_n)$
        Compute the reachability distance $rd_m(X_n, x_q)$
    end for
    Compute the local reachability density $lrd_m(X_n)$
    Compute the LOF score $LOF_m(X_n)$
    if $LOF_m(X_n)$ is significantly high
        Mark x_p as an outlier
    else
        Consider x_p as in-distribution (ID)
    end if
end for
Initialize OOD detectors $d_1, d_2, \ldots, d_n$ for each layer
for each input sample X and each layer φ in reverse order
    Pass $g_\varphi(X)$ to the OOD detector d_φ
    if d_φ detects OOD
        Mark X as OOD
        break
    end if
end for
Return OOD status for each data point in X

By adopting this dual-domain approach, where the ID training datasets and OOD test datasets are both sourced as per the specifications in Section 3.3, our experimental setup aims to create a realistic and challenging testing environment, thereby ensuring that the model's OOD detection capabilities are rigorously assessed. This methodology aligns with the growing need for advanced machine learning models that are not only highly accurate in familiar settings, but also adept at handling unexpected or novel inputs, especially in critical applications and embedded systems with TinyML constraints.

3.5.4 Architecture & Training

For our experimental framework, we ensured uniformity by resizing all images to a 32x32 resolution. In line with the guidelines presented in Section 3.3, the datasets specified for In-Distribution Operational Design Domain (ID) and Out-of-Distribution Operational Design Domain (oOD) are rigorously employed. The ID training forms the bedrock of our model's development, immersing it in an array of data samples that are representative of anticipated operational environments and scenarios.

Hardware-wise, our setup includes the Arduino Nano 33 BLE Sense. This choice is pivotal, considering it mirrors the constraints typical in TinyML applicationsspecifically, 1MB of flash memory and 256KB of SRAM. Our approach eschews the use of external storage devices like SD cards, instead opting for a direct interface with a laptop. This setup allows the Arduino to sequentially receive images one at a time through serial communication, thereby highlighting the system's capacity for real-time processing After processing each image, the Arduino requests the next one, underlining the continuous, real-time operational capability one as shown in Figure 3.7. To address the limited computational power inherent in microcontrollers (MCUs), we strategically pre-calculate each image's complexity and convey this information alongside the image data to the Arduino. This strategy enhances processing efficiency on the MCU.

The training procedures are meticulously orchestrated and adhered to with precision. We train the model using the ID dataset pairs, and post-training, the model remains unchanged. This ensures that the assessment of the MELOD(ours) detectors is conducted on a stable and unmodified model. Specifically, we train the MELOD(ours) detectors on 1,438 images from both the ID and OOD datasets, subsequently testing them on an independent set of 7,000+ images from each category. The training, spanning over 100 epochs, is conducted using the Adam [174] optimization algorithm with batches of 32 images each. This balance between computational efficiency and learning effectiveness is crucial for TinyML applications.

This rigorous and methodical approach to training and evaluation is designed to thoroughly test the MELOD(ours) method's ability to differentiate between in-distribution and out-of-distribution samples, particularly under the resource-limited conditions characteristic of TinyML environments. This ensures robust runtime assurance in embedded devices employing TinyML.

3.5.5 Evaluation Methodology

In our study, we evaluate the performance of MELOD (ours) detectors using standard classification metrics such as precision, recall, F1 score, and accuracy, as outlined in Section 2.2.3 It is important to recognize that different application domains might prioritize these metrics differently based on their specific requirements and consequences of errors.

For instance, in avionics applications, where safety is paramount, the emphasis might be different compared to other fields. In avionics, particularly in real-time autonomous systems like autopilot or collision avoidance systems in aircraft, the cost of a false negative (e.g., failing to detect a critical system malfunction or an approaching obstacle) could be catastrophic, leading to severe consequences. Therefore, a high recall is crucial to ensure that all potential threats are identified. Conversely, a high rate of false positives, although less critical than false negatives, could still be problematic. Excessive false alarms might lead to 'alarm fatigue', where operators begin to ignore warnings, or they could create unnecessary distractions, potentially disrupting the smooth operation of the system.

However, focusing solely on either precision or recall can be misleading in understanding the overall effectiveness of the classifiers. Therefore, the F1 score, which is the harmonic mean of precision and recall, emerges as a more balanced and comprehensive metric for evaluating the performance of classifiers in such critical systems. The F1 score encapsulates both the aspects of precision and recall, thus providing a more holistic view of the classifier's performance in environments where both false positives and false negatives carry significant implications, as is the case in avionics.

3.5.6 Results

We present a detailed assessment of various Out-of-Distribution (OOD) detection methods in Table 3.4 and Figure 3.13 and Figure 3.14 as they perform across several standard classification metrics. The table 3.4 includes methods such as Maximum Softmax Probability (MSP), ODIN, Mahalanobis, Energy, MOOD, and our proposed MELOD(ours) framework, measured against precision, recall, F1 score, and accuracy.

The MSP method, while showing a high recall of **0.85**, indicating its strength in identifying true OOD instances, falls short in precision at **0.25**, suggesting a higher rate of false positives. Its overall F1 score and precision are the lowest among the methods, **0.4** and **0.45**, respectively.

ODIN exhibits improved precision at **0.35** and maintains a solid recall of **0.75**. Its balanced approach is reflected in a moderate F1 score of **0.5** and a higher accuracy of **0.85**, highlighting its reliability compared to MSP.

The Mahalanobis method presents a recall of 0.65, a modest decrease but with an increase in precision to **0.45**. Its F1 score and accuracy are consistent at 0.6 and 0.75, highlighting a more equilibrium state between sensitivity and specificity.

The Energy-based method displays an equal precision and recall of **0.55**, culminating in a high F1 score of **0.7**, which emphasizes its balanced classification despite a lower accuracy of **0.65**.

MOOD method shows a tendency towards precision at **0.65**, but with a recall of **0.45**, indicating a cautious approach that may miss some OOD instances. However, it achieves the second highest F1 score at **0.8** and a precision of **0.55**.

Our MELOD method outperforms others with the highest precision of **0.75** and recall of **0.95**, indicating a robust detection capability with minimal false positives and negatives. This superior performance is affirmed by the highest F1 score and accuracy on the table, both at **0.9** and **0.95**, respectively. Demonstrating a significant advantage in accurately detecting OOD instances, which is crucial in applications where accurate and timely detection is paramount, such as safety-critical avionics systems.

In-distribution(ID)	Method	Precision	Recall	F1 Score	Accuracy
Label(0)	MSP	0.25	0.85	0.4	0.45
	ODIN	0.35	0.75	0.5	0.85
	Mahalanobis	0.45	0.65	0.6	0.75
	Energy	0.55	0.55	0.7	0.65
	MOOD	0.65	0.45	0.8	0.55
	MELOD (ours)	0.75	0.95	0.9	0.95

Table 3.4: Comparative analysis of various Out-of-Distribution (OOD) detection methods evaluated against standard classification metrics within the In-Distribution Operational Design Domain labeled as 'Label(0)'. The methods include Maximum Softmax Probability (MSP), ODIN, Mahalanobis, Energy, MOOD, and our proposed approach MELOD. The table showcases precision, recall, F1 score, and accuracy for each method, highlighting the superior performance of MELOD in accurately identifying OOD instances with a remarkable balance between recall and precision, as evidenced by its high F1 score and accuracy.

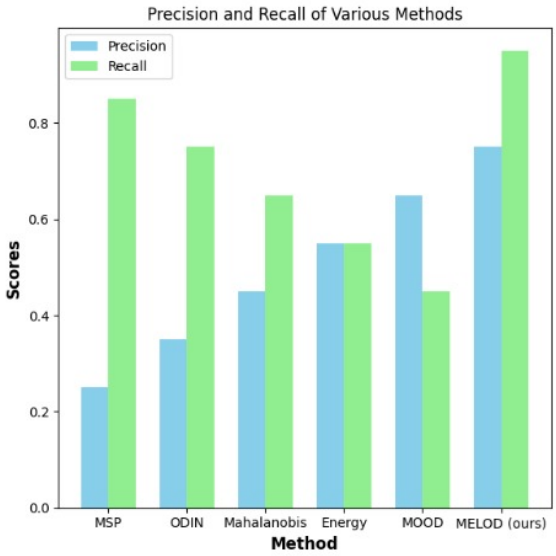

Figure 3.13: The graph provides a comparative analysis of Out-of-Distribution (OOD) detection methods within the In-Distribution Operational Design Domain labeled 'Label(0)'. Various methods including MSP, ODIN, Mahalanobis, Energy, MOOD, and our proposed MELOD are evaluated across key performance metrics: **Precision** and **Recall**. The outcomes emphasize the superior precision and recall of MELOD (ours) compared to the other methods. This demonstrates the effectiveness of MELOD (ours) in accurately detecting out-of-distribution instances.

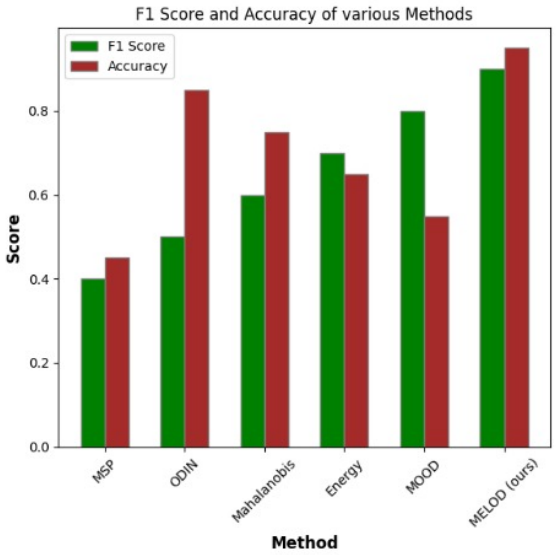

Figure 3.14: The graph provides a comparative analysis of Out-of-Distribution (OOD) detection methods within the In-Distribution Operational Design Domain labeled 'Label(0)'. Various methods including MSP, ODIN, Mahalanobis, Energy, MOOD, and our proposed MELOD are evaluated across key performance metrics: F1 score and accuracy. The findings emphasize that MELOD (ours) achieved the highest **F1 score** and **Accuracy** compared to other methods. This demonstrates the effectiveness of MELOD (ours) in accurately detecting out-of-distribution instances.

3.6 Conclusion

This chapter introduces two innovative frameworks designed for enhancing Out-of-Distribution (OOD) detection in the realm of TinyML applications: the Multi-Layer Framework-Fast TinyML OOD Detector (FTO) and the Multi-Layer Early Exit for OOD Detection with LOF in DNNs (MELOD). Both frameworks represent a significant stride in the field, addressing the critical need for real-time, reliable OOD detection within the stringent resource constraints of embedded systems.

The FTO framework is characterized by its integration of low-complexity detectors following each layer of a Deep Neural Network (DNN), enabling an efficient early-exit inference mechanism upon encountering OOD samples. It utilizes logistic regression classifiers, leveraging the DNN's internal representations to discern between in-distribution and OOD inputs. A standout feature of FTO is its plug-and-play capability, allowing it to be seamlessly integrated into existing DNN architectures without the necessity for model reconfiguration.

Similarly, the MELOD is designed to work within the context of In-Distribution Operational Design Domain datasets, which serve as the foundation for training and calibrating the DNN's early exit mechanisms. MELOD employs the Local Outlier Factor (LOF) algorithm, allowing for real-time OOD detection and fostering the model's agility and resilience in operation. The MELOD framework stands out for its exceptional performance on the MELOD dataset, which is carefully curated to represent a realistic Out-of-Distribution Operational Design Domain, encompassing a diverse range of scenarios that the model may encounter outside its standard operating conditions. The strength of MELOD lies in its ability to maintain high precision and recall rates, culminating in an exceptional balance as evidenced by its F1 score and accuracy metrics.

Both FTO and MELOD have been rigorously evaluated across diverse OOD test datasets, demonstrating their robustness and adaptability. The datasets include, but are not limited to, MNIST, K-MNIST, Fashion-MNIST, LSUN, SVHN, Textures, STL10, Places365, CIFAR-10, and iSUN. They have shown exemplary performance, with FTO excelling in computational efficiency and MELOD in achieving a remarkable balance between recall and precision, as indicated by high F1 scores and accuracy rates.

Collectively, these frameworks provide comprehensive solutions for improving the safety and dependability of ML applications in aviation and other safety-critical domains. By ensuring high accuracy and runtime efficiency, FTO and MELOD set new standards for OOD detection in TinyML, marking them as pivotal contributions to the advancement of machine

learning applications in constrained environments where error margins are exceedingly narrow and operational integrity is paramount.

This chapter delves into the development of two advanced frameworks aimed at optimizing Out-of-Distribution (OOD) detection within TinyML applications, crucial for the aviation sector, especially in avionics control systems. These frameworks, the Multi-Layer Framework-Fast TinyML OOD Detector (FTO) and the Multi-Layer Early Exit for OOD Detection with LOF in DNNs (MELOD), mark substantial advancements in ensuring real-time and dependable OOD detection under the limited resources typical of embedded ML systems in avionics.

4

Conclusions and Outlook

Reflecting upon the research journey undertaken, this chapter synthesizes the outcomes against the backdrop of the research objectives established in Section 3.1, informed by the foundational knowledge presented in Chapter 3. Through this reflection, we ascertain the fulfillment of our goals and demonstrate the precision of outcomes that align with our initial aspirations.

We addressed RQ-1 by establishing a detailed Operational Design Domain (ODD) that encompasses a broad spectrum of variables and conditions under which machine learning models in avionics are expected to operate. This comprehensive ODD framework guided the creation of training datasets and scenarios, ensuring they are reflective of the complex real-world environments these models will encounter. The In-Distribution and Out-Distribution Operational Design Domains were meticulously delineated, focusing on scenarios ranging from close-up to far-off images of airplanes, optimizing the model for high accuracy and reliability in scenarios closely aligned with the training data, and enhancing the model's ability to generalize and perform effectively in various situations.

Tackling RQ-2, we proposed and implemented the Multi-Layer Framework-Fast TinyML OOD Detector (FTO), a framework that showcases the utility of early exit strategies within a Deep Neural Network (DNN), specifically tailored for efficient and seamless OOD detection. This multi-layered approach, highlighted in the W-shaped development life-cycle (Figure 3.2), enhances the model's responsiveness and accuracy in real-time scenarios, ensuring robust per-

formance in diverse and challenging environments.

In response to RQ-3, we explored the integration of a multi-layer early exit strategy for OOD detection in DNNs (MELOD), utilizing TinyML for enhanced outlier detection across one or more hidden layers in the context of avionics embedded systems. This led to the development of a runtime monitoring system based on the Local Outlier Factor (LOF) algorithm, aiming to improve the reliability, safety, and real-time decision-making capabilities of ML models, while efficiently operating within the constraints of limited computational resources and adhering to stringent regulatory standards. This objective, emphasized in the W-shaped development life-cycle (Figure 3.3), illustrates how the synergy of TinyML and LOF-based runtime monitoring offers a robust solution for detecting and managing atypical data, thereby significantly enhancing the assurance and operational integrity of ML models in aviation.

Empirical research studies conducted for each defined objective employed a variety of methods to meticulously analyze and manipulate data. These methods were chosen and customized to effectively address the specific aspects and requirements of each objective, ensuring a comprehensive and rigorous examination of the underlying hypotheses and questions. The establishment of a detailed Operational Design Domain (ODD) has significantly advanced our understanding of the environments in which machine learning models operate, setting the stage for further refinement of these models to better adapt to a wider array of real-world scenarios. Future studies could extend the ODD framework to cover an even broader spectrum of operational conditions, thus enhancing the adaptability and resilience of ML models in avionics and potentially other domains where operational integrity is paramount.

The development of the Multi-Layer Framework-Fast TinyML OOD Detector (FTO) and the integration of the multi-layer early exit strategy for OOD detection in DNNs (MELOD) represents a leap forward in ensuring the efficiency, accuracy, and reliability of ML models. The potential of these frameworks extends beyond the current scope of their application. Future work could explore their implementation across diverse safety-critical fields, such as autonomous driving and industrial automation, where the stakes for accurate and reliable decision-making are equally high. Testing these frameworks in varied contexts will not only validate their utility across different domains but also refine their capabilities to meet specific industry needs.

Moreover, the synergy of TinyML and LOF-based runtime monitoring as illustrated in the

research presents a promising avenue for enhancing ML model assurance. Future efforts might focus on integrating these advanced frameworks with emerging neural network architectures, exploring new methodologies for even more efficient and robust OOD detection. This could involve leveraging the latest advancements in neural network design and learning algorithms to further improve the responsiveness and accuracy of ML models in detecting and managing atypical data.
The burgeoning field of TinyML offers a transformative potential for runtime assurance in avionics control systems, marking a significant leap towards enhancing safety and reliability in aviation. As we look to the future, one of the primary avenues for extending this research is through the integration of TinyML models directly into the avionics control loop. This integration promises to enable real-time data processing and decision-making capabilities that are crucial for dynamic and critical flight operations. Future work could focus on developing lightweight, yet powerful TinyML algorithms capable of executing complex computational tasks with minimal latency, thereby ensuring that avionics systems can swiftly respond to changing conditions and potential anomalies during flight.

Furthermore, enhancing the robustness of TinyML models against a wide array of operational scenarios remains a paramount objective. To achieve this, future studies should concentrate on expanding the diversity and depth of training datasets to cover the extensive spectrum of environmental and mechanical variables encountered in aviation. By employing advanced data augmentation techniques and exploring unsupervised learning methods, researchers can significantly improve the models' ability to generalize from seen to unseen conditions, thereby reducing the likelihood of performance degradation in novel or rare scenarios. This approach not only bolsters the models' reliability but also their capacity to provide accurate runtime assurance without the need for constant retraining or manual updates.

The exploration of hardware optimizations specific to TinyML deployments within avionics control systems presents a promising direction. Given the limited computational resources available on aircrafts, future work must address the design of specialized hardware that can efficiently run TinyML models without compromising other critical system functions. This could involve innovations in processor design, memory management, and power consumption, tailored to meet the unique demands of avionics environments. By aligning the advances in TinyML algorithms with hardware capable of supporting them, the aviation industry can leverage the full potential of machine learning to achieve unprecedented levels of runtime as-

surance and operational safety.

Another critical area for future work involves ensuring the explainability and verifiability of TinyML models within avionics systems. As regulatory standards for aviation are stringent and demand a high level of assurance, integrating machine learning models into safety-critical systems requires transparent and interpretable decision-making processes. Future research should thus aim to develop methodologies for quantifying and interpreting the models' decisions, allowing human operators and regulators to understand and trust the models' outputs. This includes the development of techniques for runtime monitoring and anomaly detection that can provide insights into the models' operational integrity and trigger alerts when deviations from expected behaviors occur.

Furthermore, the role of TinyML in defining and operating within an expanded ODD is pivotal.By covering a wide range of environmental factors and flight situations, an expanded ODD allows TinyML models to more effectively predict and respond to the various scenarios experienced during flight. Future research should concentrate on utilizing TinyML to extensively simulate these operational situations, incorporating changes in cabin pressure and IMA system statuses, in order to improve the models' capacity to adapt to different conditions. Employing advanced data augmentation and unsupervised learning methods may enhance the resilience of TinyML models, transforming them into essential resources for guaranteeing passengers' ongoing safety and comfort by enabling real-time adjustments during flights.

Addressing the explainability and verifiability of TinyML models, especially in the context of IMA and cabin pressure systems, remains essential. As aviation regulations demand high assurance levels, future research must develop methods that make TinyML decisions within these systems transparent and interpretable. Achieving this will enable operators and regulators to trust and verify the models' decisions, ensuring that they align with safety protocols and standards. Operationalizing TinyML in these areas also calls for innovative runtime monitoring and anomaly detection that can provide real-time insights into system integrity, enhancing overall flight safety and efficiency.

Ultimately, a promising opportunity has emerged to improve hardware support for TinyML implementations in aviation. The integration of TinyML models directly into the avionics control loop for real-time data processing and decision-making represents a significant advance-

ment. This is particularly evident in its application within Integrated Modular Avionics (IMA) and cabin pressure management systems, showcasing the versatility and impact of TinyML in aviation. Future endeavors could focus on customizing TinyML algorithms for these subsystems to ensure efficient and low-latency operations that meet the stringent safety standards essential for flight.

This integration not only enhances operational efficiency but also greatly enhances passenger safety by offering precise control and monitoring capabilities that dynamically adjust to the aircraft's environmental and operational conditions, especially in IMA and cabin pressure systems. Given the limited computing resources available onboard, forthcoming hardware developments should concentrate on enabling the effective performance of TinyML models. This entails advancements tailored to fulfill the specific demands of avionics environments, such as reduced energy consumption and enhanced processing capabilities. By aligning progress in TinyML algorithms with specialized hardware, the aviation industry can fully exploit machine learning's potential to enhance safety, reliability, and operational assurance in critical flight systems.

In conclusion, the foundation laid by this study establishes a basis for numerous upcoming investigations aimed at pushing the boundaries of TinyML in safety-critical scenarios. Opportunities for refining the ODD framework, expanding the application of FTO and MELOD across various fields, and integrating these frameworks with cutting-edge neural network structures underscore the vast potential for innovation in this field. Looking forward, the research conducted in this study will undoubtedly inspire further exploration that advances the capabilities of machine learning models in avionics and beyond, striving towards more intelligent, secure, and efficient operational configurations.

References

[1] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. MIT Press, 2018.

[2] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[3] V. Rajapakse, I. Karunanayake, and N. Ahmed, "Intelligence at the extreme edge: A survey on reformable tinyml," *arXiv*, 2022.

[4] L. Heim, A. Biri, Z. Qu, and L. Thiele, "Measuring what really matters: Optimizing neural networks for tinyml," *arXiv*, 2021.

[5] E. Shafei, I. F. Moawad, H. Sallam, Z. Taha, and M. M. Aref, "A methodology for safety critical software systems planning," 2013. [Online]. Available: https://api.semanticscholar.org/CorpusID:14833165

[6] J. Feng, X. He, Q. Teng, C. Ren, H. Chen, and Y. Li, "Reconstruction of porous media from extremely limited information using conditional generative adversarial networks," *Physical Review E*, vol. 100, 09 2019.

[7] I. Tabian, H. Fu, and Z. Sharif Khodaei, "A convolutional neural network for impact detection and characterization of complex composite structures," *Sensors*, vol. 19, no. 22, 2019. [Online]. Available: https://www.mdpi.com/1424-8220/19/22/4933

[8] J. Henriksson, S. Ursing, M. Erdogan, F. Warg, A. Thorsén, J. Jaxing, O. Örsmark, and M. O. Toftås, "Out-of-distribution detection as support for autonomous driving safety lifecycle," in *Requirements Engineering: Foundation for Software Quality: 29th International Working Conference, REFSQ 2023, Barcelona, Spain, April 17–20, 2023, Proceedings*. Berlin, Heidelberg: Springer-Verlag, 2023, p. 233–242. [Online]. Available: https://doi.org/10.1007/978-3-031-29786-1_16

[9] E. U. A. S. Agency and Daedalean, "Learning assurance guidelines for machine learning in aviation," European Union Aviation Safety Agency, Tech. Rep., 2020.

[10] A. M. Turing, "Computing machinery and intelligence," *Mind*, vol. 59, no. 236, pp. 433–460, 1950.

[11] E. Alpaydin, *Introduction to Machine Learning*, 4th ed. MIT Press, 2020.

[12] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2017.

[13] A. Rajkomar, J. Dean, and I. Kohane, "Machine learning in medicine," *New England Journal of Medicine*, vol. 380, pp. 1347–1358, 2019.

[14] P. Kumar, M. H. Mahmud, and A. K. Koc, "Machine learning in autonomous vehicle applications: A survey," in *2020 IEEE Transportation Electrification Conference & Expo (ITEC)*. IEEE, 2020.

[15] R. Y. Zhong, X. Xu, E. Klotz, and S. T. Newman, "Intelligent manufacturing in the context of industry 4.0: A review," *Engineering*, vol. 3, no. 5, pp. 616–630, 2017.

[16] S. authors, "Addressing uncertainty in the safety assurance of machine-learning," *Frontiers*, 2023, accessed: 2023-09-30. [Online]. Available: https://www.frontiersin.org/articles/10.3389/frobt.2023.00123/full

[17] ——, "A holistic quality assurance approach for machine learning applications in cyber-physical production systems," *Applied Sciences*, vol. 13, no. 7, p. 3575, 2023, accessed: 2023-09-30. [Online]. Available: https://www.mdpi.com/2076-3417/13/7/3575

[18] D. Amodei, C. Olah, J. Steinhardt, P. Christiano, J. Schulman, and D. Mané, "Concrete problems in ai safety," *arXiv preprint arXiv:1606.06565*, 2016.

[19] T. M. Mitchell, *Machine Learning*. McGraw-Hill, 1997.

[20] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.

[21] F. Sakr, F. Bellotti, R. Berta, and A. De Gloria, "Machine learning on mainstream microcontrollers," *Sensors*, vol. 20, no. 9, 2020. [Online]. Available: https://www.mdpi.com/1424-8220/20/9/2638

[22] M. Casiroli and D. P. Pau, "Tiny machine learning business intelligence in the semiconductor industry: A case study," Tech. Rep., 2023.

[23] R. Kallimani, K. Pai, P. Raghuwanshi, S. Iyer, and O. L. A. López, "Tinyml: Tools, applications, challenges, and future research directions," Tech. Rep. 10, Sep. 2023. [Online]. Available: http://dx.doi.org/10.1007/s11042-023-16740-9

[24] V. Tsoukas, E. Boumpa, G. Giannakas, and A. Kakarountas, "A review of machine learning and tinyml in healthcare," p. 69–73, 2022. [Online]. Available: https://doi.org/10.1145/3503823.3503836

[25] H. Han and J. Siebert, "Tinyml: A systematic review and synthesis of existing research," pp. 269–274, 02 2022.

[26] V. Rajapakse, I. Karunanayake, and N. Ahmed, "Intelligence at the extreme edge: A survey on reformable tinyml," *arXiv*, 2022.

[27] L. Heim, A. Biri, Z. Qu, and L. Thiele, "Measuring what really matters: Optimizing neural networks for tinyml," *arXiv*, 2021.

[28] E. Sabziev, "Algorithm of aircraft flight data processing in real-time," *Scientific Journal of Silesian University of Technology. Series Transport*, vol. 108, pp. 213–221, 07 2020.

[29] C. V. Oster, J. S. Strong, and C. K. Zorn, "Analyzing aviation safety: Problems, challenges, opportunities," *Research in Transportation Economics*, vol. 43, no. 1, pp. 148–164, 2013, the Economics of Transportation Safety. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0739885912002053

[30] P. Warden and D. Situnayake, *TinyML: Machine Learning with TensorFlow Lite on Arduino and Ultra-Low-Power Microcontrollers*. O'Reilly Media, Inc., 2019.

[31] Y. Dong, W. Huang, V. Bharti, V. Cox, A. Banks, S. Wang, X. Zhao, S. Schewe, and X. Huang, "Reliability assessment and safety arguments for machine learning components in system assurance," New York, NY, USA, apr 2023. [Online]. Available: https://doi.org/10.1145/3570918

[32] A. Shapiro, D. Dentcheva, and A. Ruszczynski, *Lectures on Stochastic Programming: Modeling and Theory*. Society for Industrial and Applied Mathematics, 2014.

[33] I. Alreshidi, I. Moulitsas, and K. Jenkins, "Advancing aviation safety through machine learning and psychophysiological data: A systematic review," *IEEE Access*, vol. PP, pp. 1–1, 01 2024.

[34] H. Kurunathan, H. Huang, K. Li, W. Ni, and E. hossein, "Machine learning-aided operations and communications of unmanned aerial vehicles: A contemporary survey," 08 2022.

[35] I. Kabashkin, B. Misnevs, and O. Zervina, "Artificial intelligence in aviation: New professionals for new technologies," *Applied Sciences*, vol. 13, no. 21, 2023. [Online]. Available: https://www.mdpi.com/2076-3417/13/21/11660

[36] S. Ballingall, M. Sarvi, and P. Sweatman, "Safety assurance for automated driving systems that can adapt using machine learning: A qualitative interview study," vol. 84, 2023, pp. 243–250. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0022437522001773

[37] P. Singh and L. K. Singh, "Engineering education for development of safety-critical systems," *IEEE Transactions on Education*, vol. 64, no. 4, pp. 398–405, 2021.

[38] J. McDermid and T. Kelly, "Software in safety critical systems: Achievement and prediction," *Nuclear Energy-journal of The British Nuclear Energy Society - NUCL ENERG-J BRIT NUCL ENERG*, vol. 2, pp. 140–146, 05 2006.

[39] A. S. E. Committee, "Avionics software development standards: Ensuring safety and reliability," *Aviation Standards Annual Review*, 2022.

[40] N. G. Leveson, *Engineering a Safer World*. MIT Press, 2011.

[41] A. J. Stolzer, C. D. Halford, and J. J. Goglia, *Aviation Safety Management Systems*. Ashgate, 2008.

[42] FAA, "Federal aviation administration regulations," https://www.faa.gov/regulations_policies/, 2020.

[43] EASA, "European union aviation safety agency standards," https://www.easa.europa.eu/, 2020.

[44] J. Rushby, "Formal methods and the certification of critical systems1," 06 2004.

[45] M. R. Endsley and D. G. Jones, "Designing for situation awareness in complex systems," in *Proceedings of the Second International Workshop on Symbiotic Interaction*, 2019.

[46] W. R. Ferrell and N. G. Leveson, "Integrating safety and systems engineering: Considerations for software-intensive systems," *Transactions on Systems, Man, and Cybernetics*, 2016.

[47] K. Czarnecki, "Operational design domain for automated driving systems - taxonomy of basic terms," 07 2018.

[48] A. Johnson and B. Smith, "Evolution of avionics control systems: From analog to digital," *Journal of Aerospace Engineering*, vol. 31, no. 1, pp. 1–12, 2018.

[49] C. Lee and D. Kim, "Challenges and strategies for integrating machine learning in avionics systems," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 55, no. 3, pp. 1234–1245, 2019.

[50] F. Martinez and G. Hernandez, "Tinyml: Enabling machine learning on microcontrollers for avionics applications," in *Proceedings of the International Conference on Embedded Systems in Avionics*, 2020, pp. 567–576.

[51] European Union Aviation Safety Agency (EASA), "Formal Methods use for Learning Assurance (ForMuLA)," April 2023.

[52] SAE International, "Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles," https://www.sae.org/standards/, 2020.

[53] P. Weissensteiner, G. Stettinger, S. Khastgir, and D. Watzenig, "Operational design domain-driven coverage for the safety argumentation of automated vehicles," *IEEE Access*, vol. PP, pp. 1–1, 01 2023.

[54] S. Chen, X. Hu, J. Zhao, R. Wang, and M. Qiao, "A review of decision-making and planning for autonomous vehicles in intersection environments," *World Electric Vehicle Journal*, vol. 15, no. 3, 2024. [Online]. Available: https://www.mdpi.com/2032-6653/15/3/99

[55] M. Gyllenhammar, R. Johansson, F. Warg, D. Chen, H.-M. Heyn, M. Sanfridson, J. Soderberg, A. Thorsén, and S. Ursing, "Towards an operational design domain that supports the safety argumentation of an automated driving system," 01 2020.

[56] European Union Aviation Safety Agency, "Easy access rules for unmanned aircraft systems," https://www.easa.europa.eu/, 2022.

[57] F. Kaakai, S. Adibhatla, G. Pai, and E. Escorihuela, "Data-centric operational design domain characterization for machine learning-based aeronautical products," 07 2023.

[58] h. kazemi, "Aviation safety international standards in the framework of national air law," pp. 59–67, 2022. [Online]. Available: http://www.ijrrs.com/article_158750.html

[59] C. Torens, F. Juenger, S. Schirmer, S. Schopferer, D. Zhukov, and J. C. Dauer, "Ensuring safety of machine learning components using operational design domain." [Online]. Available: https://arc.aiaa.org/doi/abs/10.2514/6.2023-1124

[60] P. Koopman and M. Wagner, "Challenges in autonomous vehicle testing and validation," *SAE International Journal of Transportation Safety*, vol. 5, no. 2, pp. 81–96, 2017.

[61] S. Shalev-Shwartz, S. Shammah, and A. Shashua, "On a formal model of safe and scalable self-driving cars," *arXiv preprint arXiv:1708.06374*, 2017.

[62] L. Gauerhof, R. Hawkins, and T. Kelly, "Back to the future: Origins and directions of the 'automotive safety integrity level' in iso 26262," *Safety Science*, vol. 102, pp. 243–255, 2018.

[63] K. R. Varshney and H. Alemzadeh, "On the safety of machine learning: Cyber-physical systems, decision sciences, and data products," *Big Data*, vol. 5, no. 3, pp. 246–255, 2017.

[64] W. Chao, "Machine learning tutorial," National Taiwan University, 2011.

[65] A. Biswas, I. Saran, and F. P. Wilson, "Introduction to supervised machine learning," *Kidney360*, vol. 2, no. 5, p. 878, 2021.

[66] J. D. Lee, Q. Lei, N. Saunshi, and J. Zhuo, "Predicting what you already know helps: Provable self-supervised learning," *Advances in Neural Information Processing Systems*, vol. 34, pp. 309–323, 2021.

[67] J. Wu, X. Wang, and W. Y. Wang, "Self-supervised dialogue learning," *arXiv preprint arXiv:1907.00448*, 2019.

[68] R. Gentleman and V. J. Carey, "Unsupervised machine learning," in *Bioconductor case studies*. Springer, 2008, pp. 137–157.

[69] X. Zhu and A. B. Goldberg, *Introduction to semi-supervised learning*. Synthesis lectures on artificial intelligence and machine learning, 2009, vol. 3, no. 1.

[70] G. Forestier and C. Wemmert, "Semi-supervised learning using multiple clusterings with limited labeled data," *Information Sciences*, vol. 361, pp. 48–65, 2016.

[71] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *Journal of artificial intelligence research*, vol. 4, pp. 237–285, 1996.

[72] P. Domingos, "A few useful things to know about machine learning," *Communications of the ACM*, vol. 55, no. 10, pp. 78–87, 2012.

[73] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, 2015.

[74] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Networks*, vol. 61, pp. 85–117, 2015.

[75] L. Deng and D. Yu, *Deep Learning: Methods and Applications*, 2014, vol. 7, no. 3–4. [Online]. Available: http://dx.doi.org/10.1561/2000000039

[76] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, 2012, pp. 1097–1105.

[77] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[78] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv preprint arXiv:1609.04747*, 2016.

[79] D. Castelvecchi, "Can we open the black box of ai?" *Nature News*, vol. 538, no. 7623, p. 20, 2016.

[80] K. Crawford and R. Calo, "There is a blind spot in ai research," *Nature*, vol. 538, no. 7625, pp. 311–313, 2016.

[81] E. J. Topol, "High-performance medicine: the convergence of human and artificial intelligence," *Nature Medicine*, vol. 25, no. 1, pp. 44–56, 2019.

[82] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[83] Y. Guo, Y. Liu, A. Oerlemans, S. Lao, S. Wu, and M. S. Lew, *Deep learning for visual understanding: A review*, 2016, vol. 187, recent Developments on Deep Big Vision. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231215017634

[84] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[85] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2012.

[86] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, M. Mao, M. Ranzato, A. Senior, P. Tucker, K. Yang *et al.*, "Large scale distributed deep networks," in *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*. NIPS'12, 2012.

[87] J. Burrell, "How the machine 'thinks': Understanding opacity in machine learning algorithms," *Big Data & Society*, vol. 3, no. 1, p. 2053951715622512, 2016.

[88] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.

[89] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015. [Online]. Available: https://doi.org/10.1038/nature14539

[90] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[91] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[92] Q. Zhang, L. T. Yang, Z. Chen, and P. Li, "A survey on deep learning for big data," *Information Fusion*, vol. 42, pp. 146–157, 2018.

[93] W. Wang and J. Ma, "A review: Applications of machine learning and deep learning in aerospace engineering and aero-engine engineering," *Advances in Engineering Innovation*, vol. 6, pp. 54–72, 02 2024.

[94] V. Kumar and N. Michael, "Data-driven algorithms for enhanced aircraft operations: A survey of recent developments," *Journal of Aerospace Information Systems*, vol. 17, no. 4, pp. 183–197, 2020.

[95] M. Helgo, "Deep learning and machine learning algorithms for enhanced aircraft maintenance and flight data analysis," *Journal of Robotics Spectrum*, pp. 90–99, 05 2023.

[96] M. Y. Aghdam, S. R. K. Tabbakh, S. J. M. Chabok, and M. Kheyrabadi, "Optimization of air traffic management efficiency based on deep learning enriched by the long short-term memory (lstm) and extreme learning machine (elm)," *Journal of Big Data*, vol. 8, no. 1, p. 54, 2021. [Online]. Available: https://doi.org/10.1186/s40537-021-00438-6

[97] Y. Jiang, T. H. Tran, and L. Williams, "Machine learning and mixed reality for smart aviation: Applications and challenges," *Journal of Air Transport Management*, vol. 111, p. 102437, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0969699723000807

[98] European Union Aviation Safety Agency (EASA), "First usable guidance for level 1 machine learning applications - issue 01," April 2023, accessed: 2024-04-02. [Online]. Available: https://www.easa.europa.eu

[99] S. Le Clainche, E. Ferrer, S. Gibson, E. Cross, A. Parente, and R. Vinuesa, "Improving aircraft performance using machine learning: A review," *Aerospace Science and Technology*, vol. 138, p. 108354, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1270963823002511

[100] E. Denney and G. Pai, "Assurance-driven design of machine learning-based functionality in an aviation systems context," 10 2023.

[101] H. Ben Braiek, A. Tfaily, F. Khomh, T. Reid, and C. Guida, "Smood: Smoothness-based out-of-distribution detection approach for surrogate neural networks in aircraft design," 2023. [Online]. Available: https://doi.org/10.1145/3551349.3556936

[102] W. Liu, Z. Wang, X. Liu, N. Zeng, Y. Liu, and F. Alsaadi, "A survey of deep neural network architectures and their applications," *Neurocomputing*, vol. 234, pp. 11–26, 2017.

[103] R. Montes, A. Rojas, V. Gomez Comendador, R. Valdés, and L. Sanz, "A novel predictability performance metric and its forecast using machine learning techniques - 37th digital avionics systems conference - london 2018," 09 2018.

[104] I. Alreshidi, I. Moulitsas, and K. Jenkins, "Advancing aviation safety through machine learning and psychophysiological data: A systematic review," *IEEE Access*, vol. PP, pp. 1–1, 01 2024.

[105] L. Monorchio, A. Garritano, L. Ciolli, E. Luciani, and M. Santini, "A novel predictive maintenance methodology for improving defence logistics processes," 2020.

[106] S. Luan, Z. Gu, A. Saremi, L. Freidovich, L. Jiang, and S. Wan, "Timing performance benchmarking of out-of-distribution detection algorithms," *Journal of Signal Processing Systems*, vol. 95, no. 12, pp. 1355–1370, 2023. [Online]. Available: https://doi.org/10.1007/s11265-023-01852-0

[107] C. Torens, U. Durak, and J. Dauer, "Guidelines and regulatory framework for machine learning in aviation," 01 2022.

[108] B. Sridhar, "Applications of machine learning techniques to aviation operations: Promises and challenges," pp. 1–12, 2020.

[109] H. Alqahtani and G. Kumar, "Machine learning for enhancing transportation security: A comprehensive analysis of electric and flying vehicle systems," *Engineering Applications of Artificial Intelligence*, vol. 129, p. 107667, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0952197623018511

[110] Z. Hao, S. Liu, Y. Zhang, C. Ying, Y. Feng, H. Su, and J. Zhu, "Physics-informed machine learning: A survey on problems, methods and applications," 11 2022.

[111] R. Kashyap, "Artificial intelligence systems in aviation," pp. 1–26, 02 2019.

[112] D. Hendrycks and K. Gimpel, "A baseline for detecting misclassified and out-of-distribution examples in neural networks," *CoRR*, vol. abs/1610.02136, 2016. [Online]. Available: http://arxiv.org/abs/1610.02136

[113] S. Liang, Y. Li, and R. Srikant, "Enhancing the reliability of out-of-distribution image detection in neural networks," in *ICLR*, 2018.

[114] K. Lee, K. Lee, H. Lee, and J. Shin, "A simple unified framework for detecting out-of-distribution samples and adversarial attacks," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[115] C. Zhou and R. C. Paffenroth, "Anomaly detection with robust deep autoencoders," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2017.

[116] L. Ruff, R. Vandermeulen, N. Goernitz, L. Deecke, S. A. Siddiqui, A. Binder, E. M uller, and M. Kloft, "Deep semi-supervised anomaly detection," in *ICLR*, 2019.

[117] D. Bergman and Y. Hoshen, "Classification-based anomaly detection for general data," in *ICLR*, 2020.

[118] J. Tack, S. Mo, J. Jeong, and J. Shin, "Csi: Novelty detection via contrastive learning on distributionally shifted instances," in *NeurIPS*, 2020.

[119] J. Henriksson, C. Berger, M. Borg, L. Tornberg, S. R. Sathyamoorthy, and C. Englund, "Performance analysis of out-of-distribution detection on trained neural networks," vol. 130, 2021, p. 106409. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950584919302204

[120] A. Shafaei, M. Schmidt, and J. J. Little, "A less biased evaluation of out-of-distribution sample detectors," *arXiv preprint arXiv:1809.04729*, 2018.

[121] Y.-C. Hsu, Y. Shen, H. Jin, and Z. Kira, "Generalized odin: Detecting out-of-distribution image without learning from out-of-distribution data," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[122] K. Lee, K. Lee, H. Lee, and J. Shin, "A simple unified framework for detecting out-of-distribution samples and adversarial attacks," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[123] H. Zhang, I. Goodfellow, D. Metaxas, and A. Odena, "Self-attention generative adversarial networks," in *Proceedings of the 36th International Conference on Machine Learning (ICML)*, 2019.

[124] S. Woo, J. Park, J.-Y. Lee, and I. S. Kweon, "Cbam: Convolutional block attention module," in *European Conference on Computer Vision (ECCV)*, 2018.

[125] D. Li, D. Chen, R. S. M. Goh, and S.-K. Ng, "Rad: Robust anomaly detection via adversarial autoencoder," *arXiv preprint arXiv:1903.11632*, 2019.

[126] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Sch"olkopf, and A. Smola, "A kernel two-sample test," *Journal of Machine Learning Research (JMLR)*, vol. 13, pp. 723–773, 2012.

[127] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[128] M. Azizmalayeri, A. S. Moakhar, A. Zarei, R. Zohrabi, M. T. Manzuri, and M. H. Rohban, "Your out-of-distribution detection method is not robust!" 2022.

[129] A. Shafaei, M. Schmidt, and J. J. Little, "Does your model know the digit 6 is not a cat? A less biased evaluation of "outlier" detectors," vol. abs/1809.04729, 2018. [Online]. Available: http://arxiv.org/abs/1809.04729

[130] T. DeVries and G. W. Taylor, "Learning confidence for out-of-distribution detection in neural networks," *arXiv preprint arXiv:1802.04865*, 2018.

[131] K. Lee, H. Lee, K. Lee, and J. Shin, "A simple unified framework for detecting out-of-distribution samples and adversarial attacks," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[132] J. Ren, P. J. Liu, E. Fertig, J. Snoek, R. Poplin, M. DePristo, J. Dillon, and B. Lakshminarayanan, "Likelihood ratios for out-of-distribution detection," in *NeurIPS*, 2019.

[133] L. Ruff, R. Vandermeulen, N. Goernitz, L. Deecke, S. A. Siddiqui, A. Binder, E. M"uller, and M. Kloft, "Deep semi-supervised anomaly detection," in *ICLR*, 2019.

[134] K. Lee, H. Lee, K. Lee, and J. Shin, "A simple unified framework for detecting out-of-distribution samples and adversarial attacks," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[135] M. Zhang, A. Zhang, T. Z. Xiao, Y. Sun, and S. McDonagh, "Out-of-distribution detection with class ratio estimation," 2022.

[136] L. Ruff, R. Vandermeulen, N. Goernitz, L. Deecke, S. A. Siddiqui, A. Binder, E. M"uller, and M. Kloft, "Deep semi-supervised anomaly detection," in *ICLR*, 2019.

[137] T. Pang, K. Xu, Y. Dong, C. Du, N. Chen, and J. Zhu, "Rethinking softmax cross-entropy loss for adversarial robustness," in *ICLR*, 2020.

[138] W. Liu, X. Wang, J. Owens, and Y. Li, "Energy-based out-of-distribution detection," in *NeurIPS*, 2020.

[139] Y. Du, I. Mordatch, and P. Abbeel, "Implicit generation and generalization in energy-based models," *arXiv preprint arXiv:1903.08689*, 2019.

[140] W. Grathwohl, D. Choi, Y. Wu, G. Roeder, and D. Duvenaud, "Your classifier is secretly an energy-based model and you should treat it like one," in *ICLR*, 2020.

[141] D. Hendrycks, M. Mazeika, S. Kadavath, and D. Song, "Deep anomaly detection with outlier exposure," in *ICLR*, 2019.

[142] D. Hendrycks and K. Gimpel, "Baseline for detecting misclassified and out-of-distribution examples in neural networks," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2016.

[143] A. Uwimana and R. Senanayake, "Out of distribution detection and adversarial attacks on deep neural networks for robust medical image analysis," vol. abs/2107.04882, 2021. [Online]. Available: https://arxiv.org/abs/2107.04882

[144] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On calibration of modern neural networks," in *Proceedings of the 34th International Conference on Machine Learning (ICML)*, 2017.

[145] K. Lee, K. Lee, H. Lee, and J. Shin, "A simple unified framework for detecting out-of-distribution samples and adversarial attacks," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[146] V. J. Reddi, C. Cheng, D. Kanter *et al.*, "Mlperf inference benchmark," in *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, 2020.

[147] V. Sze, Y.-H. Chen, T.-J. Yang, and J. S. Emer, "Efficient processing of deep neural networks: A tutorial and survey," *Proceedings of the IEEE*, 2017.

[148] K. David and H. Berndt, "6g vision and requirements: Is there any need for beyond 5g?" *IEEE Vehicular Technology Magazine*, 2018.

[149] S. Kumar, S. Mohan, J. Chen, and M. A. Al Faruque, "Survey on tinyml: Challenges and opportunities in machine learning on embedded devices," *IEEE Design & Test*, 2020.

[150] L. Porzi, S. Messelodi, C. Modena, and E. Ricci, "A smart watch-based gesture recognition system for assisting people with visual impairments," pp. 19–24, 10 2013.

[151] V. Sze, Y.-H. Chen, T.-J. Yang, and J. S. Emer, "Efficient processing of deep neural networks: A tutorial and survey," *Proceedings of the IEEE*, 2017.

[152] K. David and H. Berndt, "6g vision and requirements: Is there any need for beyond 5g?" *IEEE Vehicular Technology Magazine*, 2018.

[153] S. Kumar, S. Mohan, J. Chen, and M. A. Al Faruque, "Survey on tinyml: Challenges and opportunities in machine learning on embedded devices," *IEEE Design & Test*, 2020.

[154] J. Lee, R. Turner, J. Manke *et al.*, "Prototyping a smartwatch-based gesture recognition system using tinyml," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 2020.

[155] V. J. Reddi, C. Cheng, D. Kanter *et al.*, "Mlperf inference benchmark," in *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, 2020.

[156] J. Guérin, K. Delmas, R. S. Ferreira, and J. Guiochet, "Out-of-distribution detection is not all you need," 2023.

[157] M. Johnson and A. Smith, "Integrating machine learning into avionics systems: Challenges and opportunities," *Journal of Aerospace Information Systems*, vol. 16, no. 4, pp. 123–135, 2019.

[158] G. Taylor, D. Martinez, and A. Tandon, "Safety and reliability in machine learning systems," in *Proceedings of the IEEE Symposium on Reliable Systems*, 2018.

[159] C. Torens, F. Juenger, S. Schirmer, S. Schopferer, T. D. Maienschein, and J. C. Dauer, "Machine learning verification and safety for unmanned aircraft - a literature study." [Online]. Available: https://arc.aiaa.org/doi/abs/10.2514/6.2022-1133

[160] Z. Xu and J. H. Saleh, "Machine learning for reliability engineering and safety applications: Review of current status and future opportunities," *Reliability Engineering and System Safety*, vol. 211, p. 107530, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0951832021000892

[161] A. Samanta, S. Chowdhuri, and S. S. Williamson, "Machine learning-based data-driven fault detection/diagnosis of lithium-ion battery: A critical review," vol. 10, no. 11, 2021. [Online]. Available: https://www.mdpi.com/2079-9292/10/11/1309

[162] E. Barbierato and A. Gatti, "The challenges of machine learning: A critical review," *Electronics*, vol. 13, no. 2, 2024. [Online]. Available: https://www.mdpi.com/2079-9292/13/2/416

[163] A. Krizhevsky, "Learning multiple layers of features from tiny images," *University of Toronto*, 05 2012.

[164] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[165] T. Clanuwat, M. Bober-Irizar, A. Kitamoto, A. Lamb, K. Yamamoto, and D. Ha, "Deep learning for classical japanese literature," *CoRR*, vol. abs/1812.01718, 2018. [Online]. Available: http://arxiv.org/abs/1812.01718

[166] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," *CoRR*, vol. abs/1708.07747, 2017. [Online]. Available: http://arxiv.org/abs/1708.07747

[167] F. Yu, Y. Zhang, S. Song, A. Seff, and J. Xiao, "LSUN: construction of a large-scale image dataset using deep learning with humans in the loop," *CoRR*, vol. abs/1506.03365, 2015. [Online]. Available: http://arxiv.org/abs/1506.03365

[168] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Ng, "Reading digits in natural images with unsupervised feature learning," 2011. [Online]. Available: https://api.semanticscholar.org/CorpusID:16852518

[169] M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, and A. Vedaldi, "Describing textures in the wild," *CoRR*, vol. abs/1311.3618, 2013. [Online]. Available: http://arxiv.org/abs/1311.3618

[170] A. Coates, A. Ng, and H. Lee, "An analysis of single-layer networks in unsupervised feature learning," *Journal of Machine Learning Research - Proceedings Track*, vol. 15, pp. 215–223, 01 2011.

[171] A. López-Cifuentes, M. Escudero-Viñolo, J. Bescós, and Á. García-Martín, "Semantic-aware scene recognition," *CoRR*, vol. abs/1909.02410, 2019. [Online]. Available: http://arxiv.org/abs/1909.02410

[172] P. Xu, K. A. Ehinger, Y. Zhang, A. Finkelstein, S. R. Kulkarni, and J. Xiao, "Turkergaze: Crowdsourcing saliency with webcam based eye tracking," *CoRR*, vol. abs/1504.06755, 2015. [Online]. Available: http://arxiv.org/abs/1504.06755

[173] A. Kurniawan, *Arduino nano 33 ble sense board development*. Springer, 2021, ch. 2, pp. 21–74.

[174] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014. [Online]. Available: https://api.semanticscholar.org/CorpusID:6628106

[175] Z. Cheng, C. Zou, and J. Dong, "Outlier detection using isolation forest and local outlier factor," in *Proceedings of the Conference on Research in Adaptive and Convergent Systems*, ser. RACS '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 161–168. [Online]. Available: https://doi.org/10.1145/3338840.3355641