Anomaly Classification in Industrial Internet of Things Systems

Martha Lucía Rodríguez López

Tesis doctoral presentada para optar al título de Doctora en Ingeniería Electrónica y de Computación

Directores

Danny Alexandro Múnera Ramírez, Doctor (PhD) en Informática

Diana Patricia Tobón Vallejo, Doctora (PhD) en Telecomunicaciones

Universidad de Antioquia

Facultad de Ingeniería

Doctorado en Ingeniería Electrónica y de Computación

Medellín, Antioquia, Colombia

2025

Anomaly Classification in Industrial Internet of Things Systems

UNIVERSIDAD® DE ANTIOQUIA
1 8 0 3

Martha Lucía Rodríguez López

Universidad de Antioquia

This dissertation is submitted for the degree of

Doctor of Electronic and Computer Engineering

IN2LAB

Advisors

Dr. Danny Alexandro Múnera Ramírez

Dr. Diana Patricia Tobón Vallejo

Committee

Dr. Mauricio González-Palacio

Dr. Alexander Leal Piedrahita

Dr. Luis Felipe Zapata Rivera

Date

October, 2024.

Acknowledgements

This thesis would not have been possible without the support and guidance of many individuals and institutions. I want to extend my heartfelt thanks to everyone who contributed to this journey.

First and foremost, I would like to express my deepest gratitude to my two supervisors, Danny Alexandro Múnera Ramírez and Diana Patricia Tobón Vallejo. Their invaluable guidance, encouragement, and unwavering support were instrumental in shaping this research. Their insights and expertise were crucial at every stage, from the initial proposal to the final manuscript.

I want to acknowledge Universidad de Antioquia, where I pursued my PhD. The university provided a stimulating academic environment and access to numerous resources essential for my research. I am grateful for the opportunities and support provided by the engineering faculty and staff.

I am also profoundly grateful to the members of the IN2LAB research group at Universidad de Antioquia, which reviewed my thesis proposal on several occasions over these four years. Their constructive feedback and insightful suggestions during the oral presentations significantly enhanced the quality of this work. Their commitment to academic excellence has been a source of inspiration.

I would like to thank my thesis defense committee, Dr. Mauricio González-Palacio, Dr. Alexander Leal Piedrahita, and Dr. Luis Felipe Zapata Rivera, for their comments and suggestions, which allowed me to improve this work.

I also thank Université du Québec, the Institut National de la Recherche Scientifique (INRS), and Professor Long Le for hosting me during my research internship. Professor Le's advice and guidance were invaluable, and the experience greatly enriched my research.

I want to thank Servicio Nacional de Aprendizaje SENA, where I work, for their financial support and understanding. The institution's investment in my education and support enabled me to complete this doctorate.

Lastly, I am profoundly grateful to my daughter Sofía for her unwavering support,

patience, and love. Her encouragement during the challenging times and her belief in my abilities were the pillars that kept me moving forward. This thesis is dedicated to her with all my love and appreciation.

Abstract

Introducing the Industrial Internet of Things (IIoT) into traditional industrial processes brings a new age of connectivity, productivity, and insight. Through the integration of advanced sensors, communication technologies, and data analysis into industrial operations, IIoT facilitates real-time monitoring, proactive maintenance, and improved operational efficiency. Nonetheless, this heightened complexity and interconnection also bring new challenges, upholding system dependability and safety. Deviations such as unforeseen malfunctions and cyber attacks can potentially disrupt operations, resulting in substantial financial setbacks and safety risks. Considering these issues, this thesis presents an IIoT Anomaly Classification Framework designed to detect and categorize anomalies, including failures, attacks, and other significant events. The research addresses the critical need for robust anomaly detection and classification in IIoT systems by providing a comprehensive and scalable solution adaptable to various industrial contexts. The framework enhances modern industrial operations' reliability, security, and efficiency, paving the way for more resilient and intelligent IIoT systems.

The framework comprises two main components: an Anomaly Detection Model and an Anomaly Classification Model. The Anomaly Detection Model operates unsupervised, continuously monitoring system data to identify deviations from normal behavior patterns. At the same time, the anomaly classification model categorizes these anomalies based on historical data using machine learning algorithms, such as an autoencoder for the detection phase and a transformer for the anomaly classifier. This dual-model approach allows for specialized and accurate identification and categorization of anomalies, enhancing overall IIoT system reliability and security. The thesis outlines the development and implementation of the framework, including creating a comprehensive IIoT dataset and incorporating contextual information to improve detection and classification accuracy. The proposed framework has been rigorously tested in realistic IIoT environments, demonstrating its effectiveness and practicality. During the cross-validation process, a precision of 0.95, recall of 0.88, and F1-score equal to 0.91 were obtained. This research contributes significantly to IIoT, offering a valuable tool for improving industrial operations and laying the groundwork for future anomaly classification and system resilience advancements.

Contents

List of Figures

List of Tables

List of Acronyms

AI	Artificial Intelligence
A1	Data Injection Attack
A2	Denial-of-service Attack
CNN	Convolutional Neural Networks
Conv1D	Convolutional Layer 1-Dimension
CPS	Cyber-Physical System
CVS	Comma-Separated Value
F1	Temperature Failure
F2	Miscalibration Failure
F3	Disconnection Failure
FN	False Negative
FP	False Positive
GAN	Generative Adversarial Networks
GPU	Graphics Processing Unit
HMI	Human Machine Interface
HPO	Hyperparameter Optimization
IIoT	Industrial Internet of Things
IoT	Internet of Things
LSTM	Long Short-Term Memory
ML	Machine Learning
MPL	Multilayer Perceptron
MQTT	Message Queuing Telemetry Transport
NLP	Natural Language Processing
PCA	Principal Component Analysis
PLC	Programmable Logic Controller
SCADA	Supervisory Control and Data Acquisition
TN	True Negative
TP	True Positive

Chapter 1

Introduction

Integrating the Industrial Internet of Things (IIoT) into traditional industrial processes has marked the beginning of a new era of connectivity, efficiency, and intelligence [1–3]. By embedding advanced sensors, communication technologies, and data analytics into industrial operations, IIoT has enabled real-time monitoring, predictive maintenance, and enhanced operational efficiency [4–6]. However, these systems' increased complexity and interconnectivity have also introduced new challenges, particularly in maintaining system reliability and security [6]. Anomalies such as unexpected equipment failures and cyber-attacks can disrupt operations, leading to significant financial losses and safety risks [6–9].

Anomaly detection in the Industrial Internet of Things is a critical research issue due to the vulnerability of IIoT networks to novel and complex cyber threats [1]. Unlike traditional IT environments, the IIoT integrates cyber elements into core operational processes, making the consequences of security breaches more severe [10]. The dynamic and heterogeneous nature of IIoT environments, characterized by a vast array of devices and communication protocols, complicates the detection of anomalies [11]. This needs continuous research into sophisticated, adaptable anomaly detection methods that can effectively discern between normal fluctuations in data and genuine security threats [12].

The significance of advancing anomaly detection in IIoT is underscored by recent research efforts that employ diverse methodologies ranging from statistical approaches to machine learning and deep learning techniques. Key works in this area include the exploration of unsupervised techniques by Zhang et al. for multivariate time series analysis in IIoT settings [10] and the approach of Ding et al. in adapting machine learning models to handle concept drift in dynamic environments [13]. Additionally, Truong et al. introduced a lightweight architecture suitable for edge computing devices, optimizing resource usage while maintaining high detection accuracy [12]. These studies highlight the ongoing need for research that enhances detection capabilities and ensures that anomaly detection systems are scalable, efficient, and capable of operating under the constraints typical of IIoT environments.

1.1 Anomaly Classification in IIoT

The Industrial Internet of Things (IIoT) represents a significant leap in industrial automation and digital transformation. By interconnecting devices and systems through advanced networks and integrating Cyber-Physical Systems (CPS), IIoT enhances industrial operations' efficiency, productivity, and adaptability [6–9]. However, this interconnection also introduces vulnerabilities, making IIoT systems susceptible to anomalies. Effective anomaly classification is critical for maintaining the integrity, safety, and reliability of IIoT systems [6].

Anomaly classification in the Industrial Internet of Things (IIoT) remains an open research issue due to the complexities inherent in distinguishing between anomalies such as cyber-attacks and physical failures [14]. The challenge is amplified by the diversity of IIoT environments where distinct anomalies may manifest similarly, complicating accurate classification [15, 16]. For instance, Yang et al. [15] developed a data-driven approach to classify physical faults and cyber-attacks in manufacturing motor drives. Despite its effectiveness, this method relies heavily on specific motor data patterns and may not generalize across IIoT systems, highlighting the need for more adaptable and comprehensive solutions. Similarly, the solution described in Ganjkhani et al. [17] excels in identifying specific electrical faults and attack types in power systems but does not address the broad spectrum of anomalies present in IIoT domains.

Furthermore, existing solutions often focus on specific subsets of IIoT applications, which limits their applicability in a broader context. For example, the methodology by Zhang et al. [16] effectively distinguishes between sensor replay attacks and sensor bias failures in cyber-physical systems using watermarks tailored to each threat type. However, such a specialized approach may not be suitable for other IIoT applications where different underlying mechanisms might cause anomalies. Liang et al. [18] propose a scheme for multi-agent systems to minimize communication loss and enhance fault diagnosis. While this method shows promise in smart grids, its complexity and the need for extensive customization may hinder its deployment in less controlled or more heterogeneous environments. These examples underscore the persistent gap in developing effective anomaly classification systems that can dynamically adapt to the varied and evolving landscapes of IIoT, which makes continuous research and innovation necessary in this field.

1.2 Goals and Contributions

In the current state of the art, many researchers are concentrating on anomaly detection. However, anomalies must be classified to mitigate their effects on the IIoT. Since Industrial Internet of Things systems support industrial operations, they must promptly address malfunctions or cybersecurity threats. This requires the collaboration of various teams, each with specific roles: for instance, maintenance staff handles physical failures, while the IT department manages cyber attacks. Nevertheless, the detection and classification system must operate automatically with minimal human intervention to address these anomalies quickly.

The main research question in this thesis is: *How can an efficient IIoT anomaly classification framework be proposed?*

To answer this question, a series of goals were proposed that helped outline the framework for anomaly classification in IIoT. These are summarized below.

- Create an unsupervised anomaly detection model to identify deviations from normal behavior in IIoT systems.
- Develop a supervised classification model that accurately categorizes detected anomalies into specific types.
- Incorporate contextual data, such as operational conditions and historical trends, to enhance anomaly detection and classification.
- Generate a labeled dataset with a wide range of anomalies. This dataset provides a valuable resource for training and evaluating anomaly detection and classification models.
- Implement the developed framework in realistic IIoT environments and rigorously tested for effectiveness and practicality.

Considering the above, we propose an IIoT Anomaly Classification Framework to detect and categorize anomalies such as failures or attacks. This research aims to contribute significantly to the field by addressing the critical need for robust anomaly detection and classification in IIoT systems. It provides a comprehensive and scalable solution that can be adapted to various industrial contexts. The proposed framework offers a valuable tool for improving modern industrial operations' reliability, security, and efficiency, paving the way for more resilient and intelligent IIoT systems. This research makes several significant contributions to the field of IIoT:

- Comprehensive Review of Existing Research: Conducting an extensive review of current anomaly detection and classification methodologies within IIoT, identifying gaps, and highlighting opportunities for innovation.

- Creation of a Comprehensive IIoT Dataset: Providing a publicly available dataset with labeled anomalies collected from a testbed that emulates a cyber-physical system monitored by an IIoT system, including failures and cyber-attacks, will facilitate future research and development.

- Incorporation of Contextual Information: Demonstrating the value of integrating contextual data into anomaly detection and classification, leading to more accurate and reliable results and allowing the classifier to differentiate between normal CPS or IIoT events or situations caused by failures or attacks.

- Development of an IIoT anomaly Classification Framework: Introducing a new framework that combines machine learning techniques with traditional statistical methods to improve classification speed and accuracy. The goal was to create a framework that can be easily scaled and adjusted to meet the diverse requirements of various sectors.

- Real-World Testing and Validation: Implementing the proposed frameworks in emulated settings to validate their effectiveness and adaptability and conducting extensive testing and validation of the framework using a realistic testbed environment, ensuring its applicability in real-world industrial settings.

- Discuss Implications for Future Research and Practice. Anomaly classification is an ongoing process that requires continuous monitoring and model updates. Based on the implementation results, exhaustive analysis, and knowledge provided, some lines of future work are described in the field of anomaly classification in IIoT systems.

1.3 Outline of the Thesis

The rest of the thesis is structured as follows:

Chapter 2: Background. This chapter delves into anomaly classification in IIoT environments. It explores the structural complexities of IIoT architecture, detailing a layered approach that facilitates efficient data processing and analysis. By combining theoretical knowledge with practical applications, this chapter addresses the technological foundations of IIoT but also showcases innovative machine learning and deep learning techniques designed for robust anomaly detection.

Chapter 3: State of the Art. This chapter reviews the existing literature on IIoT, anomaly detection, and classification methods. It discusses the evolution of IIoT systems, the types of anomalies commonly encountered, and the current state-of-the-art techniques for anomaly detection and classification. Additionally, it highlights the challenges and limitations of existing approaches, emphasizing the need for a more robust and comprehensive framework.

Chapter 4: Framework for Anomaly Classification. It describes the methodology used in developing the IIoT anomaly classification framework. It includes detailed explanations of the data collection process, the processing techniques employed, and the selection of appropriate machine learning algorithms for anomaly detection and classification models. The chapter also discusses the integration of contextual information and the creation of the IIoT dataset.

Chapter 5: Framework Implementation Results. Presents the implementation results for the IIoT anomaly classification framework. It includes a step-by-step description of the framework's architecture, development environment, and software tools. The chapter also provides an overview of the testbed environment used for validation and the specific scenarios and use cases tested.

Chapter 6: Results. This chapter presents other results of the framework's implementation and validation. It includes detailed performance metrics, such as precision, recall, and F1-score. The chapter also includes validation results on a testbed that emulates a cyber-physical system monitored by an IIoT system. It also provides an in-depth discussion of the results, highlighting the strengths and limitations of the proposed framework and the implications of the findings for IIoT anomaly detection and classification.

Chapter 7: Conclusion and Future Work. Summarizes the key findings of the research, discusses the contributions made to the field, and outlines the proposed framework's potential impact on industrial operations. It also identifies areas for future research and development, including potential enhancements to the framework and new applications in other industrial domains.

Appendix: This includes additional data, figures, and supplementary information supporting these research findings. It describes the datasets, machine learning algorithms, and specific parameters and configurations tested. The appendix also includes code snippets and implementation details for reproducibility.

1.4 Associated Publications and Software

The state of the art presented in Chapter 3 is based on [14]:

M. Rodríguez, D. P. Tobón, and D. Múnera, "Anomaly classification in industrial internet of things: A review," Intelligent Systems with Applications, p. 200232, 2023.

Chapter 6 includes results presented at the International Engineering Congress IC-EXPOI 2022 and published in [19]:

C. EUREKA, G. de Antioquia, et al., "Engineering for transformation," in Expo Ingenieria, Fondo Editorial EIA, 2022.

Results described in Chapter 5 and Chapter 6 were published in:

M. Rodríguez, D. P. Tobón, and D. Múnera, "A Framework for Anomaly Classification in Industrial Internet of Things Systems", Internet of Things, IOT-D-24-02467.

At the time of submission of the corrected thesis, an article has been submitted for evaluation:

M. Rodríguez, D. P. Tobón, and D. Múnera, "Classification of anomalies in IIoT edge devices," Revista Facultad de Ingeniería UdeA.

The dataset used to train the models described in chapters 5 and 6 is published on https://github.com/mluciarodriguez/iiot_anomaly_classification.

The code that implements the algorithms described in chapter 5 and 6 is published on https://github.com/mluciarodriguez/iiot_anomaly_classification.

Chapter 2

Background

In the evolving landscape of industrial automation and digital transformation, the smart factory concept symbolizes a leap toward the future of manufacturing. Anchored in the principles of Industry 4.0, smart factories aim to revolutionize production processes through the seamless integration of Cyber-Physical Systems (CPS), the Industrial Internet of Things (IIoT), Big Data analytics, cloud computing, and cutting-edge artificial intelligence technologies [20]. By leveraging the IIoT, smart factories can monitor, analyze, and automate operations in real-time, significantly reducing downtime, optimizing performance, and ensuring safety, thereby leading to the creation of highly adaptive and self-optimizing production environments [10].

Within this context, a critical challenge is the effective detection and classification of anomalies, which are essential for maintaining the integrity, safety, and efficiency of these sophisticated systems [21]. Anomalies, ranging from simple equipment malfunctions to complex cyber-attacks, need quick identification and appropriate classification to prevent potential disruptions [22]. This chapter delves into anomaly detection in IIoT environments, highlighting the importance of domain-specific knowledge and classifying anomalies as failures or attacks. It further explores the structural complexities of IIoT architecture, detailing a layered approach that facilitates efficient data processing and analysis. By combining theoretical knowledge with practical applications, this chapter addresses the technological foundations of IIoT and showcases innovative machine learning and deep learning techniques designed for robust anomaly detection.

This chapter outlines a comprehensive approach to establishing and enhancing smart factory operations within Industry 4.0, particularly anomaly detection and classification within the Industrial Internet of Things (IIoT). The original contributions within this chapter are shown as follows:

- The chapter contributes to understanding how emerging technologies such as CPS, IoT, IIoT, Big Data, cloud computing, and AI are being leveraged to transform traditional manufacturing into smart factory operations. It emphasizes the role of these technologies in achieving intelligent production through interconnected manufacturing systems and vertical integration of production procedures.

- This background highlights the need for domain-specific knowledge in operating CPS and IIoT systems. It underscores the importance of expertise in distinguishing normal from abnormal operations within industrial processes.

- This chapter offers a novel classification of anomalies as events, failures, and attacks and provides a detailed blueprint for categorizing them. This classification aids in directing attention to the relevant work teams equipped with the specific domain knowledge necessary for addressing each type of anomaly.

- The delineation of the IIoT system into perception, network, and application layers contributes to structuring IIoT operations. This layered architecture simplifies the complexity of IIoT systems, ensuring efficient data processing, analysis, and storage at different hierarchical levels.

- This chapter introduces various statistical methods, machine learning algorithms, and deep learning techniques tailored for anomaly detection in high-dimensional IoT datasets. This includes the application of Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), Autoencoders, and Generative Adversarial Networks (GANs) for modeling complex relationships and temporal correlations in IIoT data.

- Addressing the challenge of unbalanced datasets in anomaly detection, the chapter outlines strategies such as resampling methods, classifier adaptation, ensemble methods, cost-sensitive approaches, and data augmentation techniques. These strategies are crucial for improving the accuracy of anomaly detection models in scenarios where the minority class holds greater significance.

- Exploring various techniques for hyperparameter optimization, including grid search, random search, Bayesian optimization, and evolutionary algorithms, significantly contributes to enhancing the performance of machine learning models in IIoT applications.

In summary, the chapter's original contributions lie in its holistic approach to integrating advanced technologies for smart manufacturing. It also contributes methodologically by outlining techniques and strategies to address common challenges in IIoT applications, such as data imbalance, anomaly classification, and hyperparameter optimization. The structure of this chapter is based on the anomaly classification framework architecture described in the introductory chapter of this document. Each section here explains the concepts associated with implementing the stages of the proposed framework for anomaly classification in IIoT systems. It begins by describing the components of a smart factory, such as CPS and IIoT. Then, some techniques for detecting and classifying anomalies are discussed in detail. Data handling before training models are discussed below. Some metrics and optimization techniques are described to close.

2.1 Smart Factory: CPS and IIoT

The smart factory aims to achieve intelligent production through interconnected manufacturing systems and vertical integration of production procedures. The smart factory utilizes automation, embedded systems, and information systems for Industry 4.0 [20]. It is a shift from traditional to smart manufacturing, supported by emerging technologies such as Cyber-Physical Systems (CPS), Internet of Things (IoT), Big Data, cloud computing, and advanced Artificial Intelligence [20]. The Industrial Internet of Things (IIoT) refers to the fusion of physical systems with digital technology. This integration enables Cyber-Physical Systems to optimize production processes, reduce downtime, improve safety, and introduce new ways to personalize production [10, 12, 23].

In this manuscript, the term cyber-physical system refers to an automated industrial process. In contrast, Industrial Internet of Things is the system that monitors the CPS through cloud applications to optimize its performance. This thesis deals with the classification of anomalies, specifically in IIoT, not in CPS, since the classification of anomalies in IIoT is a topic that has not been treated in depth in the scientific literature. This statement is developed in detail in the state-of-the-art chapter.

2.1.1 Cyber-Physical System

An industrial process is a set of organized operations or stages designed to carry out specific industrial activities. These processes involve converting raw materials, energy, or components into finished products or services [10]. A cyber-physical system (CPS) integrates computational (SCADA and PLCs) and physical processes with computer algorithms monitoring and controlling physical elements, with feedback loops where physical processes affect computations and vice versa [10, 12, 23]. Specialized knowledge is required to operate a CPS optimally. Each industrial process represents a specific domain knowledge and requires expertise or a deep understanding of a particular area, field, or discipline [24]. This knowledge encompasses the concepts, techniques, tools, and vocabulary unique to that domain. For example, the maintenance team of an industrial process must know how it works and the value ranges for the variables to distinguish normal from abnormal behavior [25].

2.1.2 Internet of Things

The Internet of Things (IoT) refers to a network of physical objects, commonly known as things, equipped with sensors, software, and other advanced technologies [1–3]. This enables them to connect and share data with other devices and systems via Internet. The most significant advantage of IoT devices is their ability to collect and transmit data, which facilitates remote monitoring, automation, and a wide range of smart applications. These applications can improve efficiency, convenience, safety, and health [4–6]. Integrating IoT devices into the web of connectivity presents various challenges, ranging from security and privacy to interoperability and standards development [6]. Nevertheless, IoT continues to grow in popularity, fueled by advancements in wireless networking technologies, the miniaturization of electronics, and the widespread availability of internet access [1, 4–6].

2.1.3 Industrial Internet of Things

For [26], IIoT is the application of IoT in industrial control systems. The Industrial Internet of Things (IIoT) is a network of connected devices and sensors in various industrial applications. Its main function is to collect and analyze real-time data, which, in turn, helps to improve efficiency, reliability, and decision-making processes [6–9]. By leveraging advanced technologies such as cloud computing, machine learning, and big data analytics, IIoT optimizes production processes, reduces operational costs, and improves product quality. Additionally, it enables higher levels of automation, predictive maintenance, and safer working environments, resulting in significant improvements in performance and productivity [9, 12, 27].

In modern smart factories, using Industrial Internet of Things technology has enabled the implementation of predictive maintenance techniques [9, 12]. By leveraging real-time data from sensors and other connected devices, predictive maintenance algorithms can detect potential machine failures before they occur. This allows for proactive maintenance and repairs, reducing downtime and increasing equipment lifespan. Ultimately, this approach can lead to increased efficiency, reduced costs, and improved safety in manufacturing operations [6–8, 27].

IIoT Architecture

An Industrial Internet of Things (IIoT) system is designed with a layered approach to handle the complexity and scale of industrial operations efficiently. This architecture integrates various devices, platforms, and services [4, 5, 7, 8]. Although there is no consensus on the exact number of layers in the architecture, it can be streamlined into three primary layers. These layers are responsible for device integration, data processing, and application services, ensuring seamless operation. Here is a brief overview of these layers,

- **Perception Layer**: Smart devices, sensors, and actuators are incorporated into the industrial equipment to create a foundational layer. This layer performs data collection directly from the environment. The devices in this layer measure various physical parameters such as temperature, pressure, and vibration. Consequently, they become the primary source of real-time data necessary for the efficient operation of the machinery [7, 9, 28].
- **Network Layer**: It receives the processed information the perception layer provides and transmits it to the application. The network layer integrates various devices (e.g., switching, gateway) and various communication technologies (e.g., Bluetooth, Wi-Fi) [28]. In this thesis, the network layer is on the edge of the IIoT system and acts as a mediator by focusing on processing local data, analyzing it, and making preliminary decisions close to the data source. Its functionalities include data processing, filtering, and minor analytics designed to reduce latency, manage bandwidth efficiently, and support immediate operational responses. Furthermore, this layer may have localized security measures to ensure data integrity at the network's edge [7, 9].
- **Application Layer**: The Cloud Layer is the topmost level in this architecture and is responsible for processing, analyzing, and storing massive amounts of data.

It hosts IIoT platforms with advanced analytics, machine learning capabilities, and comprehensive data management tools. This layer enables long-term data analysis, trend prediction, and strategic decision-making. Additionally, it supports device management and application deployment while integrating security protocols across the entire IIoT system [7, 9, 28].

Figure 2.1 shows the IIoT Architecture suggested in this thesis. It comprises three layers: Perception, Network, and Application. The schematic illustrates IIoT separated from the Cyber-Physical System.

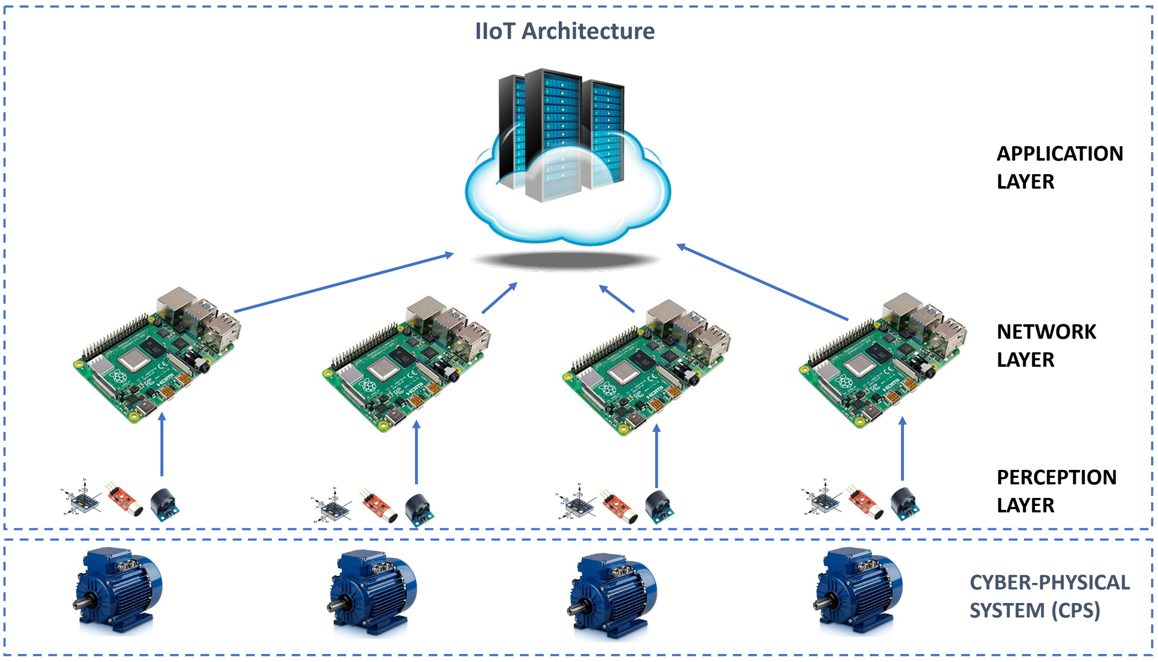

Figure 2.1: IIoT Architecture comprises three layers: Perception, Network, and Application. The schematic illustrates IIoT separated from the Cyber-Physical System.

2.2 Anomalies in IIoT Systems

Implementing an IIoT system poses some notable challenges, including security issues, unusual behavior, and service interruptions. It remains a challenge to tackle security and resilience concerns, which calls for a focus on improving security measures [22]. By detecting anomalies, we can gain insights and prevent potential risks. Anomaly detection identifies some rare values that deviate significantly from the majority of data [21].

An anomaly refers to any value significantly different from most of the remaining values in a dataset [29, 30]. Anomalies can arise from a variety of factors, including malicious attacks, sensor faults, and major environmental changes, as recorded by the sensors [31–33]. The causes of these anomalies can vary, but their effects are not necessarily distinct. For example, a malicious attack can be launched to take advantage of IIoT network's computing power or to damage the system by generating false data. In the former case, the attack could manifest as excess erroneous network traffic, while in the latter, it could appear as false messages passed off as legitimate [11, 34].

2.2.1 Anomaly Types

The concept of anomalies in datasets has been widely discussed in the literature, with three main types identified: point anomalies, contextual anomalies, and collective anomalies [4, 32, 34]. On the one hand, point anomalies refer to values that significantly deviate from other values in a dataset, representing a single point of interest [11]. On the other hand, contextual anomalies are values that may or may not be considered outliers depending on the context in which they appear. A value considered an outlier in one context may not be in another. Finally, collective anomalies are groups of related values that largely differ from the other values in the dataset [4, 30, 34, 35].

This thesis uses the classification given by Ghosh et al. in [31]. They classify the anomalies as events, failures, and attacks. This classification helps direct the attention of the relevant enterprise work teams with specific domain knowledge. For instance, the maintenance staff can resolve device failures, while the IT team can counter cyber attacks. This research reserves the term 'event' for samples initially classified as anomalies that are subsequently labeled as normal by the classifier.

Saurav et al. [36] defined an **anomaly** as a behavior that deviates from the norm. While Ghosh et al. [31] refers to an outlier as an observation or a subset of observations that deviate significantly from the rest of the dataset. Additionally, the latter also defines event, failure, and attack:

- For Ghosh et al., an **event** refers to an occurrence that causes a change in the current state of the environment. This could be a natural phenomenon that affects the values of monitored variables. Such anomalies typically persist and change the pattern of data. However, in this thesis, the term 'event' will be used to label a sample classified as 'abnormal' by the anomaly detector and, subsequently, classified as 'normal' by the anomaly classifier. This contradiction will make it necessary for a human operator to manually diagnose the sample's label.

However, it can be challenging to differentiate between an error caused by faulty sensors and an event since both can cause prolonged anomalies. Spatial correlation is an essential tool for identifying the type of anomaly since measurements from faulty sensors lack spatial correlation, unlike data from events.

- A **failure** happens when sensor data is incorrectly measured due to a device malfunction or a lack of calibration. Fault-induced outliers are more common than those resulting from events, and they differ from other data because they arise randomly. These errors can compromise data quality, making it necessary to identify and remove them to ensure efficient use of data.

- In a sensor network or IIoT devices, an **attack** refers to the malicious infiltration of one or more nodes. This can deceive other nodes into participating and compromise the entire network's security. The most common method of system intrusion is wireless communication.

Differentiating between failure and attack is necessary for reliable data and preventing malicious attacks [29]. Industrial control systems based on IIoT require a lightweight, accurate anomaly detection solution to identify complex threats quickly [11, 12, 37].

2.2.2 Anomaly Detection

Anomaly Detection (AD) is the process of identifying data points, events, or observations that deviate significantly from the normal behavior of a dataset. These anomalies are often called outliers and can signify critical incidents such as system failures or security breaches [10, 12, 13, 33]. In industrial settings, AD solution monitors equipment to predict and prevent failures. The main objective of anomaly detection is to uncover these irregularities that traditional statistical methods might not identify due to their rarity or unusual nature [34, 37]. The anomaly score is the prediction error for each timestamp, and instances, where the score exceeds the threshold, are classified as anomalies [33, 37].

Accurately defining what constitutes normal behavior is the key to successful anomaly detection. However, this can be challenging as normal behavior can vary significantly across different datasets and contexts. Various techniques can be used for anomaly detection, ranging from simple statistical methods to complex machine-learning algorithms. The latter can learn the definition of *normal* from the data, making it an effective way to detect anomalies [33, 34].

2.2.3 Anomaly Classification

Industrial Internet of Things encompasses a network of interconnected devices and machines that communicate with each other and human users, enabling advanced monitoring, analysis, and automation of industrial processes [16]. Classifying anomalies in the IIoT is vital for several reasons, all of which contribute to industrial operations' enhanced efficiency, safety, and reliability of predictive maintenance, quality control, safety and compliance, operational efficiency, resource optimization, cybersecurity, customization, and adaptation [38, 39]. In the context of Industrial Internet of Things, identifying anomalies is critical in preventing adverse outcomes and enabling positive improvements in industrial operations. To achieve this, advanced machine learning and data analytics technologies are being increasingly deployed to enhance anomaly classification in IIoT. This makes it a vital area of focus for industries looking to leverage the full potential of digital transformation [38, 39].

2.2.4 Anomaly Detection and Classification Techniques

Detecting anomalies has become necessary in data analytics and is widely investigated in statistics and machine learning [9, 32]. The prediction-based method predicts future time series using the fitted model on the training data. It detects anomalies based on the difference between the predicted and actual value [9, 37]. Anomaly detection and classification in the Industrial Internet of Things (IIoT) are pivotal for maintaining industrial systems' integrity, safety, and efficiency [9, 32]. Most supervised learning techniques that use binary classifiers to detect anomalies can be used as multi-class classifiers to identify the anomaly that affects the IIoT [9, 37]. Below is a comprehensive overview of these techniques.

Statistical Methods

Statistical Methods set thresholds for expected behavior based on historical data [9, 32, 40].

Autoregressive models help detect anomalies by assigning an anomaly score based on the difference between the actual value and the predicted one [41]. The most common types of autoregressive models for univariate time series are Autoregression (AR), Moving Average (MA), Autoregressive Moving Average (ARMA), and Autoregressive Integrated Moving Average (ARIMA). For multivariate time series, the Vector Autoregression (VAR) model is a common type of regression model [35, 40–42].

Control charts are tools used for statistical process control. They help monitor the shift in mean and variance over time series data. The primary objective of control charts is to identify whether the variation in the process is caused by natural variations (background noise) or some external factor. When the process is under control, it indicates the presence of natural variations. On the other hand, when the process is out of control, it shows the occurrence of special variations [35, 43].

Spectral techniques estimate the data by merging attributes representing most data variability. These techniques assume the data can be embedded in a lower-dimensional subspace, where normal instances and anomalies appear significantly different [44]. Two popular methods in this category are Principal Component Analysis (PCA) and Singular Spectrum Analysis (SSA) [35].

Machine Learning Algorithms

Machine learning is crucial in anomaly detection in Industrial Internet of Things. Anomaly detection aims to identify data points, events, or observations that deviate significantly from the dataset's normal behavior. Machine learning models can automate and enhance the accuracy of detecting these anomalies [2, 9, 32, 35, 40]. Here's an overview of how machine learning is applied in anomaly detection,

Supervised Learning. Techniques such as Decision Trees (DT), Support Vector Machines (SVM), and Gradient Boosting Machines (GBM) are utilized when historical data is available as labeled normal or anomalous [9, 32]. Tree-based methods split data into non-overlapping leaves using tree structures, making them particularly effective for high-dimensional and non-gaussian data distributions [35].

Unsupervised Learning. Clustering is a type of unsupervised machine learning that involves dividing data points into several clusters to maximize the similarity between points within the same cluster and minimize the similarity between different clusters. The effectiveness of clustering is determined by two criteria: maximum inter-cluster distance and minimum intra-cluster distance [35, 45–47]. Algorithms such as K-means clustering and Principal Component Analysis (PCA) find patterns or groupings in data without pre-labeled classes, identifying anomalies as outliers.

Density-Based Spatial Clustering of Applications with Noise (DBSCAN) is a widely used unsupervised clustering approach that requires little domain expertise and can handle relatively large samples. It works because a point is considered a part of a cluster if it is closely situated to many other points from the same cluster. Low-density areas often separate high-density regions [35].

Semi-Supervised Learning. In semi-supervised algorithms, the model is trained only on majority class data. The assumption is that the class of interest is either unknown or too rare to include in training. The model learns only the majority class behavior and predicts whether a new data point fits that behavior. Semi-supervised learning is used when a mix of a small amount of labeled data and a larger volume of unlabeled data improves the model's detection capabilities [23].

Deep Learning

Recent improvements in classification for high-dimensional IoT datasets are based on deep learning techniques. These approaches model complex, highly nonlinear interrelationships between multiple sensors and efficiently capture temporal correlation [35]. Complex algorithms that capture high-level features in datasets are built with neural networks [33, 37]:

Convolutional neural networks (CNNs) are feed-forward deep neural networks initially developed for image analysis but are now also used for processing multidimensional time series and effectively extracting correlations. In CNNs, convolution operations are used in at least one layer to extract patterns from the underlying spatiotemporal structure of the time series [33, 35, 37].

Recurrent Neural Networks. One popular method for classification in time series data using deep neural networks involves regression techniques to forecast one or more future values based on past values. The difference between the predicted and actual values is then calculated to determine if the predicted point is in a class or not [35]. Recurrent Neural Networks (RNNs), Long Short-Term Memory (LSTM), and Gated Recurrent Units (GRU) networks are examples of this method for time-series data [33, 37].

A **Multilayer Perceptron** (MLP) is a class of feedforward artificial neural network (ANN) that consists of at least three layers of nodes, such as an input layer, one or more hidden layers, and an output layer. Unlike single-layer perceptrons, which can only learn linear functions, MLPs can learn to model non-linear relationships due to their multiple layers and the use of non-linear activation functions. This makes MLPs a powerful tool for many machine learning tasks, including classification, regression, and even feature learning in deep learning contexts [33, 35, 37].

Transformer is a powerful Natural Language Processing (NLP) model designed for translation tasks. However, it has recently demonstrated impressive performance in time series processing [48]. One of the significant contributions of Transformer is its self-attention mechanism incorporated into the neural network [49]. A typical Transformer model comprises two primary components: the encoder and the decoder. On the one hand, the encoder processes the input sequence and transforms it into a continuous representation that holds all the information learned from the input. On the other hand, the decoder produces the output sequence based on the encoded information and all previously generated elements of the output [10, 11, 13, 49, 50].

Generative Adversarial Networks (GANs) is a machine learning framework developed in 2014 by Ian Goodfellow and his team [51]. These frameworks consist of two neural networks - a generator and a discriminator - that are trained concurrently in a competitive manner [48, 51]. The training method is a zero-sum game in which the generator attempts to increase the probability of the discriminator making a mistake. In contrast, the discriminator seeks to distinguish real data from fake data better. This game continues until the discriminator can no longer easily differentiate between real and generated data, indicating that the generator has learned to create highly realistic data [48, 51].

Unsupervised learning techniques like GANs have been used to detect time series data anomalies. To detect anomalies, GAN-based methods primarily concentrate on generating distinctive features by utilizing adversarial training on normal samples. This results in the features of the abnormal samples being significantly different from those of normal samples, and this difference is reflected in the reconstruction error and discrimination value [30].

Ensemble Methods. Combining multiple models to improve anomaly detection accuracies, such as Random Forests and ensembles of autoencoders, where predictions from several models are aggregated [23, 46].

Federated learning is a machine learning approach that enables a model to be trained across multiple decentralized devices or servers holding local data samples without exchanging them. This technique is beneficial for privacy preservation, data security, and reducing the need for data centralization and transmission. It is particularly relevant in scenarios where data privacy is paramount, such as healthcare, finance, and mobile device usage [46, 47, 52]

Autoencoders are artificial neural networks used for dimensionality reduction or feature learning. They encode an input into a lower-dimensional form and then reconstruct the output from this representation [53]. They consist of an encoder and a decoder, which compress input data and regenerate it from the latent space representation [9, 53]. Autoencoders are trained by minimizing the reconstruction error, which is the difference between the original input and its reconstruction [53]. This encourages the network to learn useful data properties, making autoencoders useful for anomaly detection, denoising, and generative models [9]. Another class of Autoencoders is the Variational Autoencoders (VAEs), which replace the reconstruction error with the reconstruction probability. VAEs are unsupervised deep-learning generative methods that use Bayesian inference to model the training data distribution [33, 35, 37].

Threshold Values. A threshold value distinguishes normal and unusual data points. An assigned score is used to differentiate between the two. Techniques balancing sensitivity and specificity are used to calculate the best threshold value. Common approaches are used for autoencoders in anomaly detection [53, 54].

- Fixed Threshold. A predefined threshold value is often based on domain knowledge or empirical observation. You can also take the maximum value of the residual as a threshold when the model is evaluated with normal data [9].
- Mean Plus Standard Deviation. Set the threshold as the mean reconstruction error of the training data plus a multiple of the standard deviation. This method assumes that the normal data's reconstruction errors follow a Gaussian distribution [53].
- Iterative Method. To find the best threshold value for a given metric, start with a low threshold and increase it by a specific amount on each iteration. Then, calculate the metric to optimize, such as F1-score, Precision-Recall Curve, or Receiver Operating Characteristic (ROC) Curve, and choose the threshold that gives the best metric value [40].
- Dynamic Thresholding. In dynamic environments, a fixed threshold may not be effective. Adaptive thresholding techniques adjust the threshold based on changing data patterns, recent performance, and feedback [53–55].

2.3 Data for Anomaly Classification in IIoT

Accurate data from smart devices is crucial for the success of the Industrial Internet of Things. It helps monitor and optimize systems, anticipate potential issues, and make informed decisions to enhance productivity and efficiency. This section will discuss aspects that must be considered when selecting the variables and processing the data before using them to train the anomaly detection and classification models.

2.3.1 Feature Selection

Currently, a lot of industrial equipment is fitted with multiple sensors that keep track of the equipment's condition. This produces a significant amount of time series data with numerous variables. However, identifying irregularities in this data is difficult because of the complex temporal relationship between each variable and the correlations between different variables. Although numerous algorithms have been proposed, only a small number have effectively tackled these challenges when identifying anomalies in multivariate time series data [10]. The excessive and unnecessary features present in data with high dimensions lead to computational complexity and significantly decrease the precision and effectiveness of classification techniques [2, 56]. Uninformative features can be divided into low variance, highly correlated, and highly collinear. Low variance features have slight variation among samples and should be removed. Highly correlated features show similar patterns among samples and can complicate the learning process. Highly collinear features have cause-effect relationships and can also complicate learning [2, 57].

Utilizing unprocessed time series data as input for training might not be the most effective for industrial applications. Therefore, they suggest that extracting or choosing features from time series data through statistical techniques or human expertise could significantly simplify the dataset [30]. Dimensionality reduction techniques can decrease the number of features that describe an object, making it easier to understand anomalies [29]. Different feature selection techniques can be used for IIoT security, including gini-impurity, chi-square, information gain, mutual information, and feature correlation [2]. Feature correlation selects informative features based on a given threshold range. This technique has achieved competitive detection accuracy while maintaining a reasonable level of complexity compared to other selection methods. A correlation-based feature selection approach calculates the correlation between each feature and the target variable. The Pearson correlation coefficient, Spearman index, and mutual information technique are commonly used to implement this approach. The goal is to select features that are not correlated to one another to reduce multicollinearity [2]. Pearson correlation is used when a linear relation exists between variables; the Spearman index is helpful with no linear relations, and mutual information determines the correlation between random features.

Dimensionality reduction techniques can be divided into two main categories: feature extraction and feature selection. On the one hand, feature extraction involves reducing the dimensionality of data by projecting the original features into a new low-dimensional feature space [2, 56]. Two standard methods used for feature extraction are principal component analysis and linear decision analysis. On the other hand, feature selection involves selecting a subset of features from the original feature space that is more effective in predicting class labels and has a lower feature dimensionality. There are four categories of feature selection methods: wrapper, filter, hybrid, and embedded methods. Wrapper methods evaluate the predictive accuracy of a chosen learning algorithm to determine the quality of a selected feature subset. Filter methods, in contrast, are not reliant on predetermined learning algorithms. Hybrid methods are a combination of both filter and wrapper methods. Finally, embedded methods incorporate feature selection into the learning algorithm's training process [2, 56].

Data with high dimensions can be problematic due to excessive and unnecessary features. These features can lead to computational complexity, reducing the precision and effectiveness of classification techniques [2, 56]. Researchers often perform feature extraction or selection to effectively train their anomaly detection models. In Chapter 3, you can find a description of the most commonly used methods for reducing the dimensionality of a dataset. However, some authors have developed their methods.

Authors in [58] describe a statistical methodology used for dimensionality reduction in time series called Symbolic Essential Attributes Approximation (SEAA). This approach helps to simplify multidimensional time series data by creating a new nominal representation of the original data series. SEAA is based on the concept of data series envelopes and essential attributes generated by a multilayer neural network. Meanwhile, MIMIC-FS is a feature selection method for processing information on the Internet of Things. This method employs a statistical measure known as the maximal information coefficient (MIC) to determine the relevance and redundancy of features to class labels. The MIC can capture various types of correlations between variables, including linear, nonlinear, and nonfunctional correlations. This makes the feature selection process more accurate and efficient [56].

Additionally, [59] introduced a wrapper-based algorithm called CorrAUC to select useful features for machine learning algorithms. The method uses the area under the curve (AUC) metric to test its effectiveness. While [60] designed a method for selecting features in industrial control systems using a genetic algorithm. This method eliminates redundant features by utilizing a mechanism for feature ranking fusion in the genetic algorithm. Furthermore, the proposed method incorporates the growing tree clustering idea to enhance the global merit-seeking speed.

2.3.2 Context Information

The status of an environment is determined by various contextual factors such as users, applications, locations, or devices. To ensure interoperability, pervasive computational systems need to have the ability to share context, which helps them maintain a shared understanding of contextual information. This information is closely linked to data humans can easily understand while reading. Therefore, sharing contextual information is essential in facilitating efficient communication between different systems and stakeholders involved [61].

Any information that can be used to characterize the situation of an entity is referred to as context [62]. An entity, which can be a machine, place, or object, is considered to be relevant to the interaction between a user and an application [63]. Context-awareness is an aspect of Internet of Things applications, especially in the industrial sector. It enables providing relevant and appropriate services to users based on their understanding of the current situation and all entities involved. In simpler terms, context-awareness involves using information collected from various sources to provide services that cater to a user's specific needs and expectations [62]. Using context information extracted from the industrial process can help classify the type of anomaly in the IIoT system. For instance, sound sensors collect raw data, but interpreting this data to determine whether the motor is off or the sensor is failing is context information. Some context variables can be data from the CPS, external such as environmental, commercial, and production, or artificially created from equations and data available. Typically, the choice of context variables is made by an expert who knows the industrial process and the IIoT system.

2.3.3 Data Processing

The success of the Industrial Internet of Things relies on gathering accurate data from smart devices throughout the industrial facility. This data is used to make informed decisions and provide services. However, IIoT devices' data can often be uncertain, noisy, erroneous, voluminous, and widely scattered. Additionally, this data can exhibit gradual changes, continuity, correlations, periodic patterns, and behaviors consistent with Markovian processes [4, 5]. Poor data quality can lead to incorrect decisions, declining user engagement, and acceptance of IIoT services. Therefore, it is crucial to maintain high data quality standards to ensure widespread adoption and acceptance of the IoT paradigm and services by users [4].

Converting raw data into a structured and meaningful format suitable for analysis is called data processing [4]. This process is decisive as it ensures that data is clean, consistent, and high-quality, making it suitable for machine learning-based classification models. Data processing plays a significant role in preparing the data for analysis and ensuring its reliability [2]. The data processing includes feature elimination, missing value handling, duplicate removal, encoding of non-numerical features, and normalization [2].

Exploratory Data Analysis

When researchers begin studying a dataset, they usually conduct an Exploratory Data Analysis (EDA) as the first step. This crucial process involves using various statistical graphs and data visualization techniques to scrutinize the dataset and identify any patterns, anomalies, or features that may be present. By doing so, the researchers can test any hypotheses and verify assumptions that they have made about the data. The EDA process forms the foundation of any data-driven research, helping researchers to gain a deeper understanding of the dataset and the insights it may hold [46].

Missing Data

Missing values can cause significant gaps in the information available to the user regarding a particular entity or phenomenon. Since these datasets serve as input for knowledge derivation processes, such gaps can lead to incomplete understanding or erroneous decision-making. Therefore, missing values can harm the overall quality of the data [4, 64, 65]. Here are some strategies we can employ to handle missing data effectively:

- Delete rows or columns with missing values. Row deletion removes rows with missing values if the dataset is large and the number of rows with missing data is relatively small. Column deletion removes columns with a high percentage of missing values, especially if those columns are not critical for analysis or modeling [65].

- Impute missing values. Mean/median/mode imputation replaces missing values with the column's mean, median, or mode. This is useful for numerical data. Another strategy is forward or backward fill, which carries forward the previous data point's value or fills backward with the following data point's value. This is particularly useful in time-series data [65].

- Hot-deck imputation. It is a statistical method used for handling missing data in datasets. This technique involves filling in missing data with observed values from similar records within the same dataset. The term “hot-deck” implies that the data used to fill in the gaps are drawn from the same dataset (“deck”) and are current or “hot,” meaning they are from the same period or study context [66, 67].
- Assign a unique value to missing data, such as -1 for numeric columns or a tag like ‘unknown’ for categorical columns. This can be useful when the fact that data is missing may carry its information [65].
- Use algorithms that support missing values. Specific algorithms inherently support missing values, such as k-Nearest Neighbors (k-NN), which can ignore a missing value during the distance calculation phase, or tree-based algorithms like Decision Trees, Random Forest, and Gradient Boosting Machines [65].
- Techniques like Multiple Imputation by Chained Equations (MICE) use the entire set of available feature variables to estimate missing values. This can be more accurate than univariate methods but is computationally more intensive [65].
- Interpolation is the process of approximating missing values using other available values. Data streams refer to estimating missing attributes or tuples, which can arise from sensor malfunctions, connection loss, and related problems [4, 65]. Various methods exist for interpolating data points, such as linear and polynomial interpolation.

Data cleaning

Managing datasets involves a critical component known as data cleaning. It is not exclusive to the IoT context and comprises three key phases: identifying error types, detecting potential errors, and remedying the identified errors. Data cleaning techniques go beyond the detection of outliers and interpolation, encompassing a diverse range of activities that guarantee the quality and reliability of data [4, 64]. For example, since duplicate instances do not contribute significantly to the model creation process, they are discarded directly.

Non-Numerical Features Encoding

It is necessary to convert the non-numerical features into numerical ones so that machine learning algorithms can process the data. Label encoding and one-hot encoding are methods for handling categorical variables in machine learning [2]. The choice between them depends on the specific dataset and the ML algorithm. Label encoding is more straightforward and space-efficient but may introduce an arbitrary order to categorical values. One-hot encoding avoids this issue by creating binary columns for each category, but it can lead to high-dimensional data [2].

Normalization vs Standardization

The difference in data distribution between different features can significantly impact the classification model and feature selection process, in addition to the high computational resources required for training such a large dataset. The data are usually scaled using normalization or min-max scaling methods to solve this problem [2, 60]. The normalization equation is:

$$z = \frac{x - \mu}{\sigma} \tag{2.1}$$

Where:

- z is the normalized value.
- x is the original value.
- μ is the mean of the dataset.
- σ is the standard deviation of the dataset.

This formula adjusts values to have a mean of 0 and a standard deviation of 1, facilitating specific statistical analyses and training machine learning models.

The formula for min-max scaling is:

$$x' = a + \frac{(x - \min(x))(b - a)}{\max(x) - \min(x)} \tag{2.2}$$

Where:

- x' is the rescaled value.
- x is the original value.
- $\min(x)$ and $\max(x)$ are the minimum and maximum values in the dataset, respectively.
- a and b are the new minimum and maximum values after rescaling.

This formula adjusts the values of the data to a specific range $[a, b]$, commonly to $[0, 1]$ or $[-1, 1]$, which can help compare measures that have different units or scales.

Data Splitting

The splitting of data involves dividing the initial data into two separate sets. The first is the training set used to train the model. The second set, known as the test set, is used to assess the final performance of the trained model. To test the effectiveness of the trained model, the test data set must maintain a similar distribution of classes as the training set. This is done to simulate the actual scenario of IIoT networks [2].

Sequences of Data

The sliding window technique in time series analysis transforms a dataset into a format suitable for machine learning models, especially deep learning. Transforming time series data from *[n-samples, n-features]* to *[n-samples, timesteps, n-features]* is vital for several reasons [9, 11, 37, 48]:

- Time series problems depend on past observations to predict future values. By using a windowed format, the model can learn these dependencies. The window size determines how far back the model can look to make a prediction.
- RNNs, LSTMs, and GRUs are machine learning models for sequential data. They work by processing input data sequentially, one step at a time, while retaining hidden states that carry information across these time steps.
- To improve the performance of a model working with multivariate time series, converting the dataset to include information from all features in each time step is helpful. This allows the model to learn complex relationships between features within the same time frame.
- This conversion facilitates both single-step and multi-step forecasting. The windowed approach provides the necessary structure for both strategies.
- During training, the model is exposed to more varied sequences by presenting data in overlapping windows.
- This technique leverages existing observations to make more efficient use of available data instead of discarding valuable temporal information, creating a more extensive set of training samples.

The size of the window used in anomaly detection is vital. It should be large enough to capture essential patterns and small enough to avoid diluting the anomalies. There is no single optimal window size that works for all scenarios. It often involves trying different sizes and using domain knowledge to balance capturing relevant information and maintaining computational efficiency [11, 37, 48].

2.3.4 Balancing Data in Time Series

Unbalanced datasets in classification models are a crucial issue significantly affecting predictive model performance. This occurs when the distribution of classes is not uniform, causing bias towards the more frequent class. Building classification models for an unbalanced dataset requires special consideration for satisfactory performance [38, 68]. Imbalanced datasets can cause machine learning models to favor the majority class, leading to inadequate performance for the minority class. This is particularly problematic for applications like anomaly detection and predicting rare events where the minority class is of greater significance [38, 68, 69].

In multiclass classification, some approaches combat class imbalance: resampling, classifier adaptation, ensemble strategies, and cost-sensitive techniques. Resampling adjusts the training data distribution by oversampling the minority class or undersampling the majority class; classifier adaptation modifies the classifier's decision boundary; ensemble strategies combine multiple classifiers; and cost-sensitive techniques penalize misclassification of minority classes more heavily [38, 70]. Other methods include deep learning algorithms, data augmentation, fairness, and bias mitigation.

Resampling Methods

Resampling methods are widely used to overcome data imbalance issues in machine learning by processing the dataset to create a more balanced distribution of classes [57, 69]. These methods usually involve undersampling, which removes instances from the majority class, oversampling, which involves cases augmenting in the minority class, or a hybrid approach of both. Additionally, these approaches can be categorized as either random or heuristic techniques based on adding or removing samples [30, 38, 69]. The Synthetic Minority Over-sampling Technique (SMOTE) is an advanced method that creates synthetic samples to force the decision region of the minority class to become more general [30, 42].

Random resampling techniques can lead to the loss of crucial information or overfitting of data. On the other hand, heuristic methods offer a more refined approach by selectively removing insignificant instances from the majority class or by creating cases synthetic for the minority class. These heuristic methods often employ algorithms like SMOTE to ensure the data's integrity is maintained or improved [38].

Classifier Adaptation

One strategy to deal with imbalanced datasets is to adapt machine learning models in a way that they can better understand and learn from the class distribution. This can be achieved by developing specialized algorithms designed to accommodate the imbalance in class distributions besides the standard technique of resampling [38].

Ensemble Methods

Ensemble methods are a powerful tool for building highly accurate predictive models by combining the outputs of multiple base models [69]. While these methods have been successfully applied to single-class classification tasks, their potential in multiclass classification lies in their ability to integrate diverse predictions from multiple classifiers trained on different data segments. This not only enhances overall predictive performance but also addresses issues related to class imbalance [38, 69].

Cost-Sensitive Approaches

Cost-sensitive techniques are a popular solution for addressing the issue of class imbalance in multiclass classification settings. These techniques assign varying costs to errors, emphasizing higher costs for misclassifying minority class instances [57, 69]. By doing so, the approach seeks to minimize the overall misclassification cost, which can help achieve a more targeted and effective solution to the imbalance problem [38, 71].

Deep Learning Approaches

With the advent of deep learning, there has been a surge in using intricate neural network architectures to tackle class imbalance. This involves utilizing autoencoders to extract features and learn representations to improve the model's ability to discern significant patterns in the minority class data. Moreover, cutting-edge loss functions considering class imbalance during the training phase have been suggested to steer the model optimization process [71].

Fairness and Bias Mitigation

As machine learning models can reinforce bias, researchers have been working on developing fairness-aware algorithms to address class imbalance. The goal is to improve the model's performance without sacrificing fairness. These approaches consider both predictive performance and ethical implications to ensure that the model is unbiased and produces fair results [72].

Data Augmentation

Data augmentation techniques are advantageous for deep learning models as they prevent overfitting. One essential method to create efficient synthetic data is to simulate anomalous conditions using a model or simulation process replicating industrial processes under parameters yet to be experienced in the real world. Reliable and high-fidelity simulation data can offer low-cost training data and address the problem of insufficient samples [30].

Data augmentation can be significantly enhanced using Generative Adversarial Networks (GANs), particularly in domains where data is limited, expensive to acquire, or requires more variety than the original dataset. GANs consist of two neural networks, the generator and the discriminator, that are trained together in a competitive process. The generator produces data that is nearly identical to actual data, while the discriminator distinguishes between real and generated data [30].

2.4 Evaluation and Optimization

A model's performance is measured on specific tasks during the evaluation phase. Metrics such as accuracy, precision, recall, and F1 score are used to assess its performance. Proper evaluation techniques like cross-validation or splitting data into training and test sets can help estimate how well a model generalizes to new data. In addition, hyperparameters can be adjusted to control the learning process, and hyperparameter optimization (HPO) can help identify the most effective combination.

2.4.1 Metrics

In binary and multiclass classification, evaluation metrics are utilized to measure the distinct characteristics of the algorithm. These metrics assess the algorithm's performance through its accuracy, precision, recall, and F1 score. Precisely, accuracy measures the proportion of correctly classified instances, while precision measures the proportion of true positives among all instances classified as positive. Conversely, recall measures the proportion of true positives among all actual positive instances, and F1-score is the harmonic mean of precision and recall [34, 38, 73]. By leveraging these evaluation metrics, we can gain insights into the strengths and weaknesses of the classification algorithm and make informed decisions about its suitability for a given task. The evaluation metrics are designed to assess the generalization capability of the trained classifier, mainly when tested with previously unseen data [73].

Confusion Matrix

The confusion matrix helps determine the best solution during classification training. It arranges the predicted and actual classes in rows and columns, respectively. TP and TN represent the number of correctly classified positive and negative instances, while FP and FN denote the misclassified negative and positive instances. By analyzing these values in the confusion matrix, we can determine the most effective solution for classification training [73, 74].

	Actual: Yes	**Actual: No**
Predicted: Yes	True Positives (TP)	False Positives (FP)
Predicted: No	False Negatives (FN)	True Negatives (TN)

Accuracy

In machine learning, accuracy is a performance metric that evaluates the ability of a classifier to predict the correct outcomes of new data. It is calculated by dividing the number of correct predictions by the total number of predictions made. Accuracy measures the proportion of correctly classified instances out of all instances tested [38, 73, 74].

$$\text{Accuracy} = \frac{\text{True Positives} + \text{True Negatives}}{\text{Total Observations}} \tag{2.3}$$

False Positive Rate or False Alarm Rate

The False Positive Rate (FPR) is the proportion of negative instances wrongly classified as positive. It is used to evaluate the accuracy of binary classification models to ensure that decisions are based on reliable data [73]. FPR is calculated as follows:

$$FPR = \frac{FP}{TN + FP} \tag{2.4}$$

where:

- FP represents the number of false positives,
- TN represents the number of true negatives,
- The denominator $(TN + FP)$ is the total number of actual negatives.

The FPR is a critical measure in evaluating the specificity of a classification model. It indicates how well the model avoids false alarms.

False Negative Rate

The False Negative Rate (FNR) represents the positive instances that are inaccurately classified as negative. It is a significant factor in evaluating the accuracy of a model and is frequently used to fine-tune algorithms and improve outcomes [73]. It is calculated as follows:

$$FNR = \frac{FN}{TP + FN} \tag{2.5}$$

where:

- FN represents the number of false negatives,
- TP represents the number of true positives,
- The denominator $(TP + FN)$ is the total number of actual positive instances.

The FNR is crucial for understanding the sensitivity of a classification model. It indicates the model's effectiveness in identifying positive cases without mistakenly labeling them negative.

True Positive Rate

The True Positive Rate (TPR), or sensitivity or recall, is the proportion of actual positive instances the model correctly identifies [38, 39]. It is calculated as follows:

$$TPR = \frac{TP}{TP + FN} \tag{2.6}$$

where:

- TP represents the number of true positives,

- FN represents the number of false negatives,
- The denominator $(TP + FN)$ is the total number of actual positive instances.

The TPR assesses the model's effectiveness in identifying positive cases and is particularly critical in scenarios where failing to detect positives is costly.

True Negative Rate

The True Negative Rate (TNR), known as specificity, measures the proportion of actual negative instances correctly identified by a classification model. It is a crucial metric for assessing a model's ability to distinguish negative cases from positive ones, especially when the cost of misidentifying negatives (as positives) is high [38, 39]. It is calculated as follows:

$$TNR = \frac{TN}{TN + FP} \tag{2.7}$$

Where:

- TN represents the number of true negatives,
- FP represents the number of false positives,
- The denominator $(TN + FP)$ is the total number of actual negative instances.

The TNR is critical for evaluating the model's ability to correctly identify negative cases, ensuring it can effectively distinguish between negative and positive instances without causing unnecessary alarms.

Precision

Precision, also known as the positive predictive value, is the proportion of true positives among all instances classified as positive by the model [34, 38, 74]. It is calculated as follows:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{2.8}$$

Where:

- TP represents the number of true positives,
- FP represents the number of false positives,

- The denominator $(TP + FP)$ is the total number of instances classified as positive by the model.

Precision is crucial for understanding the reliability of a model's positive classifications, emphasizing the importance of minimizing false positive rates.

F1-Score

The F1-score is defined as the harmonic mean of precision and recall, offering a balance between the two when evaluating the accuracy of a binary classification model. It is particularly useful in scenarios with uneven class distribution [34, 38, 39, 74]. The F1-score is calculated as follows:

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2.9}$$

Where:

- Precision is the proportion of true positives among all instances classified as positive,
- Recall (or Sensitivity) is the proportion of true positives identified correctly among all actual positives,

The F1-score ranges from 0 to 1, where 1 indicates perfect precision and recall, and 0 indicates the worst.

For evaluating multiclass classification models, traditional metrics have to be redefined [34, 38, 39, 74].

1. **Accuracy**: The proportion of correctly predicted observations to the total observations. Useful but can be misleading in imbalanced datasets.
2. **Precision (Per-Class and Macro/Micro-Averaged)**: The ratio of correctly predicted positive observations to the total predicted positives, averaged across classes.
3. **Recall (Per-Class and Macro/Micro-Averaged)**: The ratio of correctly predicted positive observations to all observations in actual class, averaged across classes.
4. **F1 Score (Per-Class and Macro/Micro-Averaged)**: The weighted average of Precision and Recall, considering both false positives and false negatives.

5. **Confusion Matrix**: A table used to describe the performance of a classification model on a set of test data for which the true values are known.

6. **Matthews Correlation Coefficient (MCC)**: A correlation coefficient between the observed and predicted binary classifications, providing a balanced measure that can be used even with very different class sizes.

7. **Area Under the ROC Curve (AUC-ROC)**: The area under the ROC curve can be extended to multi-class classification through One-vs-All (OvA) schemes.

8. **Log Loss (Cross-Entropy Loss)**: Measures the performance of a classification model where the prediction input is a probability value between 0 and 1.

The metric selection depends on the particular goals of the classification task and the dataset's attributes. Each metric offers a distinct viewpoint on the model's performance. Therefore, it is key to opt for a metric that is suitable for the given task [34, 38, 39, 74]. Various classification metrics can be obtained from the confusion matrix to assess and compare the effectiveness of different machine-learning models [42]. As the evaluation datasets for anomaly detection are usually unbalanced, most authors use the standard confusion matrix metrics: true positives (TP), false negatives (FN), false positives (FP), true negatives (TN) and their derivate metrics, including accuracy, Precision, Recall, F1-score, Area under curve (AUC), and Area under Precision-Recall (AUPR) to evaluate the performance of the models [2, 3, 9–11, 13, 35, 37, 40, 47, 50, 52–54, 60, 70]

Some authors utilize certain parameters specific to the environment where the model is deployed to assess its overall performance. [46] evaluates the suitability of its FL-based models for resource-constrained environments, such as smart meters, using edge devices' memory, CPU usage, bandwidth, and power consumption as metrics. [48] evaluated the performance of their DCT-GAN model using F1-Score and AUROC (Area Under Receiver Operating Characteristic curve). Moreover, [12] utilizes precision, recall, and F1-score to evaluate edge computing performance concerning memory usage, GPU and CPU usage, power consumption, run time, and bandwidth occupation.

Researchers can also choose to create their own metrics. [73] and [75] have identified several crucial factors to consider when selecting or creating a metric for evaluating classification models. These factors include:

- **Classification Type**: The first step is identifying whether the model is intended for binary or multiclass classification. This differentiation is crucial because it influences the selection of suitable performance evaluation metrics.

- **Efficiency in Computation**: Consider using a simple measurement that requires little computational power, enabling a fast and efficient evaluation.
- **Understandability and Discriminative Power**: The metric must distinguish between varying degrees of model effectiveness to accurately identify successful models.
- **Depth of Performance Insight**: It is important to select a metric that provides a detailed comprehension of the model's effectiveness, which goes beyond a basic accuracy measure.
- **Attention to Underrepresented Classes**: Ensure that the metric used to evaluate the model's performance considers the minority classes, especially when imbalanced data distribution. This will promote fairness and accuracy in the evaluation process.

These criteria will ensure that the metric accurately measures the performance of classification models while also considering factors such as efficiency and fairness [73, 75].

2.4.2 Hyper Parameter Optimization

Optimization is an important task in anomaly classification. It involves tuning model parameters, adjusting learning rates, and selecting appropriate algorithms. This process requires precision, domain knowledge, and practical considerations. Doing so can increase the probability of correctly identifying anomalies, thereby reducing false positives and negatives. To achieve the best results, it is necessary to train models on available datasets using optimal parameters [9]. Different combinations of hyperparameters such as batch size, learning rate, dropout rate, number of epochs, optimizer, number of layers, and number of nodes are often used to train a model. Setting these hyperparameters before training the model is crucial, as they can impact the learning speed and model performance [9]. Hyperparameter optimization techniques range from manual tuning to automated methods. Commonly used techniques include:

- Grid search explores a subset of hyperparameters by training and evaluating a model for each possible combination of hyperparameter values defined in a grid. This brute-force approach systematically goes through all combinations of hyperparameter values [76–79].

- Random search is a technique that picks random combinations of hyperparameters to train and evaluate a model. This process is repeated for a specified number of iterations [76, 77].
- A probabilistic model is built by Bayesian optimization to map the function from hyperparameter values to the objective evaluated on a validation set [76, 77].
- Gradient-based optimization methods adjust hyperparameters in the direction that reduces the validation loss, based on the gradient of the loss concerning the hyperparameters [76–78].
- Mechanisms inspired by biological evolution, such as mutation, crossover, and selection, are utilized by evolutionary algorithms to evolve a population of hyperparameter sets towards better performance [76, 77].
- Hyperband is an approach that uses a bandit mechanism to allocate resources to a group of hyperparameter configurations. It works by swiftly discarding underperforming configurations while granting more resources to the ones that show promise [76–79].

Despite the availability of numerous hyperparameter optimization (HPO) software packages and GPU resources, manual experimentation remains the most commonly used approach for optimizing hyperparameters. This approach relies on a researcher's intuition, domain knowledge, and preliminary explorations, which are usually inexpensive [76, 78]. A suggestion to bridge the gap between HPO algorithms and Deep Learning (DL) researchers is PriorBand, an HPO algorithm designed explicitly for DL that enables the integration of expert knowledge and inexpensive proxy tasks. In another study, [78] suggests that the high cost of time and computing power is a significant obstacle to the widespread adoption of modern HPO practices. [78] has proposed a two-stage HPO technique to reduce the computational load of conventional HPO by using a small subset of the training dataset. The best candidates shortlisted from the first stage are retrained with the full training dataset for final selection, resulting in final models ready for deployment.

Chapter 3

State of the Art

The Industrial Internet of Things (IIoT) has brought unprecedented connectivity and data collection capabilities in manufacturing, energy, and transportation. This surge in connected devices and sensors has enabled real-time monitoring and optimization of industrial processes and exposed these systems to many potential anomalies and security threats [9, 12, 27]. Anomalies in the IIoT context can range from simple system malfunctions to sophisticated cyberattacks, each with the potential to disrupt operations and inflict significant damage. Given this backdrop, developing robust anomaly detection and classification systems becomes paramount to safeguarding the integrity and functionality of IIoT ecosystems. These systems must distinguish between normal operations and abnormal behaviors, ensuring timely and appropriate responses to potential threats.

We have developed a comprehensive review to investigate the advancements in anomaly classification methodologies within the IIoT realm. This work was originally published in Elsevier's Intelligent Systems with Applications in 2023 [14]. We systematically categorize existing research based on anomaly detection techniques, statistical methods, traditional machine learning, and advanced deep learning algorithms. Additionally, we explore the contextual application of these techniques, examining whether the strategies are context-aware and identifying the computational environments (end devices, edges, clouds, or local servers) in which these algorithms are deployed. Our analysis highlights the current research trends, reflecting a growing interest in leveraging deep learning for anomaly detection on edge devices and underscores the critical role of context-aware information in enhancing detection accuracy. By providing a structured overview of anomaly detection strategies and their implementation nuances, this state-of-the-art aims to inform and inspire future research directions that address the unique challenges posed by the IIoT landscape.

We have observed a continuous increase in published papers in recent years, indicating that this subject has sparked the research community's interest. We have also examined the techniques employed by various authors to detect anomalies and have identified whether they detect anomalies in general or specific types of anomalies such as events, failures, or attacks. Additionally, we have analyzed the use of contextual information in implementing anomaly detectors. An overview of the report's findings is presented in Figure 3.1, which displays the percentage of papers meeting each classification criterion. Out of the articles reviewed, only 8% utilized context-aware information. Most of the reports (56.5%) utilized deep learning methodologies for anomaly detection. Anomaly detection was intended for edge devices (48.9%), while the anomaly detection system was implemented primarily on the edge computing (19%) layer. Finally, most papers (58.5%) do not disclose the location where the algorithms were implemented.

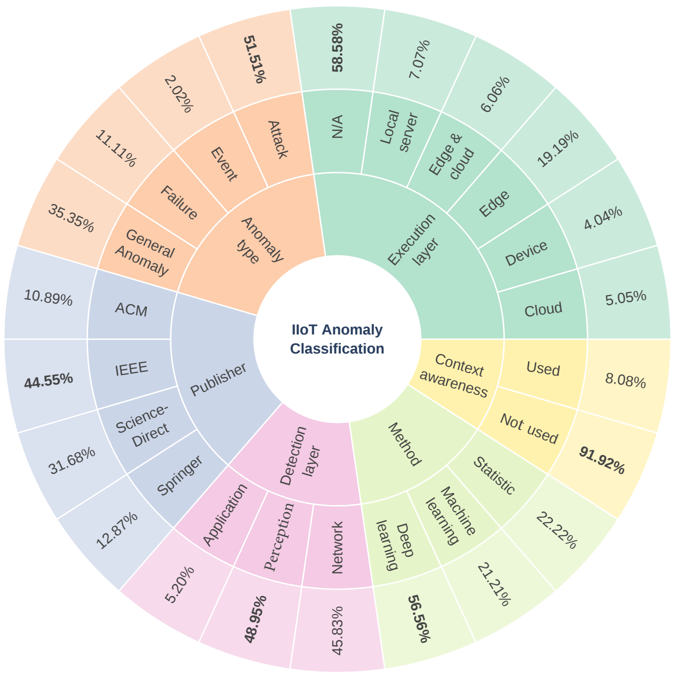

Figure 3.1: Percentage of papers included in the state-of-the-art review that meet each classification criterion.

This chapter emphasizes anomaly detection and classification advancements within the Industrial Internet of Things (IIoT). Here's a synthesis of the key original contributions:

- Comprehensive Review of Anomaly Classification Methods: This chapter classifies current research based on the methods used for anomaly detection (statistical, machine learning, deep learning) and the contexts in which they are applied. This organized categorization helps identify trends and gaps in anomaly classification within the IIoT, providing a clear overview of the field.

- Advanced Anomaly Detection Strategies: Highlighting advanced strategies like the use of autoencoders for unsupervised anomaly detection and the integration of Federated Learning for distributed anomaly detection systems emphasizes the chapter's contribution to showcasing cutting-edge methodologies adapted for the complexities of the IIoT.
- Context-aware Anomaly Detection: The emphasis on using context-aware information for enhancing anomaly detection mechanisms underscores an original contribution towards developing more intelligent, situational-aware IIoT systems.
- Identification of Trends and Gaps: The analysis highlights a continuous increase in publications, reflecting growing interest in the field. It also points out specific focus areas, such as the predominant use of deep learning methodologies and the implementation of anomaly detection on edge devices, indicating a trend towards decentralized computing.

These contributions are significant as this chapter presents a synthesis of current research trends and methodologies in anomaly detection within IIoT that could inform future research and application development in this rapidly evolving field.

3.1 Methodology

The review focused on understanding anomaly detection in IIoT and identified available solutions to classify anomalies as events, failures, or attacks. Also, it was explored how context-aware information has been used to improve anomaly detection algorithms. To elaborate on this review, we considered the guidelines for conducting systematic mapping studies in software engineering by Petersen et al. [80].

We used the guidelines on [80] to outline our review methodology. First, we formulated a set of research questions that align with the primary objective of the review. Next, we developed a search query based on these questions and applied it to relevant databases. Then, we filtered the retrieved papers using exclusion criteria to determine their eligibility. After reviewing the abstracts, we assessed whether the articles were related to anomaly detection in IIoT. Finally, we manually removed duplicate submissions by identifying and eliminating duplicate copies of articles from different journal databases. The data obtained from these steps formed the foundation of our review. Below, we provide a detailed explanation of each of these steps.

3.1.1 Research Questions

We have defined three research questions for our state-of-the-art analysis as described to follow,

- RQ1: Which techniques and methods enable detecting and classifying anomalies in IIoT?
- RQ2: What kind of validation do authors perform for proposed strategies?
- RQ3: How does context information improve anomaly detection in IIoT?

3.1.2 Search Query and Databases

We created the following search query by identifying the main keywords in the research question:

```
(("Industrial IoT" OR IIoT OR "Industrial Internet of Things") AND "anomaly
detect*")
```

Then, this query was applied to four relevant databases in the field, such as Springer, Science Direct, ACM, and IEEE.

3.1.3 Exclusion Criteria

We identified peer-reviewed and available online papers describing techniques or methods to detect anomalies in IIoT. Still, we excluded surveys, systematic reviews, mapping studies, editorials, prefaces, interviews, news, correspondences, discussions, comments, readers' letters, panel discussions, poster sessions, abstracts, or books.

3.1.4 Classification criteria

Following the defined methodology, we used the research questions to identify the criteria for extracting information from the retrieved papers, as shown below.

- Criterion 1: Does the paper detect events, failures, or attacks? This criterion indicates whether a work detects events, failures, attacks, or combinations. This information was extracted following the definition of an event, failure, and attack given in Chapter 2.
- Criterion 2: Does the work use statistical, machine learning, or deep learning methods to detect anomalies? The reviewed papers are divided into three groups: those that primarily use statistical methods to detect anomalies, those that use machine learning techniques, and those that specifically use deep learning.
- Criterion 3: Does the work diagnose anomaly origin as an event, failure, and attack? This criterion determines whether a strategy differentiates the anomaly's origin or not.
- Criterion 4: Does the work use context information? In this review, context information is additional data collected from an industrial process for anomaly detection. This criterion aims to identify whether a work uses this context information to implement the detection or classification algorithm.
- Criterion 5: Where does the work detect anomalies? in the perception, network, or application layer? This criterion indicates whether a proposal detects anomalies occurring at the perception, network, or application layer of the IIoT system.
- Criterion 6: Which device runs the anomaly detection algorithm (node, edge, cloud, or local server)? Each type of device in the IIoT network has different constrained resources (e.g., storage, processing, and latency), which limits the possibility of implementing the model in real time. The following sections show the results of applying this methodology to the selected reports.

Below are the results of applying this state-of-the-art review methodology to the selected works.

3.2 Anomaly Detection

Anomaly Detection involves identifying data points or observations that significantly deviate from the normal behavior of a dataset [10, 12, 13, 33]. There are two types of intrusion detection systems: signature-based and anomaly-based. Signature-based uses pattern matching to recognize known intrusion behaviors but has a high false negative rate. Anomaly-based needs a lot of network traffic data to detect abnormal behaviors and can detect new types of attacks, but it has a higher false positive rate [3]. Anomaly detection is a significant security concern in the Industrial Internet of Things (IIoT) due to the increased risk of cyberattacks and failures on distributed devices and critical infrastructure networks [1, 10, 11]. Several proposals have been developed to detect anomalies in IoT, IIoT, and cyber-physical systems in recent years. The works found in the review are grouped below according to the techniques used to detect anomalies, such as statistics, machine learning, and deep learning.

3.2.1 Statistical Techniques

Various statistical methods have been applied to implement anomaly detection systems, such as phase-aware hidden semi-Markov model [81], discrete wavelet transform [82], fast Fourier transform [83], swap center metric [84], singular spectrum analysis [85], time series correlation with Pearson coefficient [86], principal component analysis [87], Markov chains for discrete time [88, 89], Bayesian dynamic equalization assigning reward and punishment mechanisms to IoT nodes [90], extended Kalman filter [91], deterministic finite automata [92], Dempster-Shafer's Theory of Evidence [33, 93], linear dynamic state space models [94], and null space based on stochastic subspace identification methods [95].

We reported the information extracted from the selected papers in three Tables with a similar format (Tables 3.1-3.3). The first column of each table shows the paper's reference, and the second column briefly describes the detection strategy. The column identified as *anomaly* specifies whether the report detects events, failures, attacks, or anomalies. The *classify* column, in turn, indicates whether that research identifies anomaly sources. Based on the six criteria cited in [14], Table 3.1 summarizes the scientific papers using statistical methods to detect anomalies. The *context-awareness* column records report that use additional information (context information) taken from a process to detect anomalies, as well as the *detection layer* column indicates whether the authors detect anomalies occurring in IIoT devices, networks, or application layers. Finally, the *execute layer* column shows where the authors proposed implementing the algorithm.

It can be seen from Table 3.1 that some authors detect anomalies from different origins but without classifying the type of anomaly. For example, Garitano et al. [87] detect (man-in-the-middle) attacks, sensor failures, and communication problems by examining the contribution of each variable to an abnormal event. However, their proposal does not automatically detect the source of the anomaly, limiting the functionality of generating an alarm for a human operator who diagnoses whether it is a plant event, a communication event, or an attack. The authors include data from physical and network variables in their detection method, which could be considered context information, called "metadata."

Table 3.1: Anomaly detection proposals using statistical methods ('E' stands for Events, 'F' stands for Failures, and 'A' stands for Attacks).

Reference	**Anomaly detection strategy**	**Anomaly**	**Classify (E-F-A)**	**Context Aware**	**Detection Layer**	**Execute Layer**
(Yeganeh, 2023) [43]	Control Charts modeled by a Poisson distribution	Attacks	✗	✓	Network	-
(Chang, 2023) [44]	Spatial and Spectral maps	Attacks	✗	✓	Network	-
(Mazarbhuiya, 2023) [45]	partitioning and agglomerative hierarchical	All	✗	✓	Network	-
(Hashmat, 2022) [96]	Vulnerability signature formation engine	Attacks	✗	✓	Network	-
(Aruquipa, 2022) [97]	Bio inspired manufacturing with vibration sensors	All	✗	✗	Perception	Device
(Wang, 2022) [98]	Correlation between time series through the self-attention mechanism	All	✗	✗	Perception	-
(Zhan, 2021) [99]	Hierarchical representation for time series anomaly detection	All	✗	✗	Perception	-
(De Vita, 2021) [83]	Semi-Supervised Bayesian Anomaly Detection	Failures	✗	✗	Perception	Edge & Cloud
(Cai, 2021) [81]	Content-agnostic payload-based anomaly detector	Attacks	✗	✗	Network	-
(Dang, 2021) [82]	Discrete Wavelet Transform and Principal Component Analysis	All	✗	✗	Perception	Local server
(De Vita, 2020) [100]	On board fault prediction by analyzing real time sensor data	Failures	✗	✗	Perception	Edge & Cloud
(Gorbenko, 2020) [84]	Swap centre metric method	Attacks	✗	✗	Perception	-
(Aoudi, 2020) [85]	Singular spectrum analysis	Attacks	✗	✗	Perception	-
(Li, 2020) [86]	Correlation between multivariate time series	All	✗	✗	Perception	-
(Garitano, 2019) [87]	Monitoring incoming connection patterns on server side	Attacks	✗	✓	Network	Local server
(Faisal, 2019) [88]	Deep-packet inspection	All	✗	✗	Network	-
(Wang, 2020) [101]	Dynamic Bayesian equalization	Attacks	✗	✗	Network	-
(Bernieri, 2019) [91]	Extended Kalman Filter (EKF)	Attacks	✗	✗	Perception	Device
(Genge, 2019) [89]	Hotelling's T2 statistics and the univariate cumulative sum	All	✗	✗	Perception	-
(Bernieri, 2019) [92]	Deterministic Finite Automata	Attacks	✗	✗	Network	-
(Enuachescu, 2019) [93]	Dempster-Shafer's "Theory of Evidence"	Attacks	✗	✗	Perception	-
(Peng, 2019) [102]	Fuzzy theory	Events	✗	✓	Application	Edge & Cloud
(Ghaeini, 2018) [94]	Linear Dynamical State-space (LDS)	Attacks	✗	✗	Perception	Local server
(Zugasti, 2018) [95]	Stochastic Subspace Identification	Attacks	✗	✗	Application	-
(Madhawa, 2018) [103]	Invariants are formulated by experts	Attacks	✗	✗	Perception	Device

Ghaeini et al. [94] present an approach to detect anomalies, focusing specifically on malicious attack detection. On the other hand, Hashmat et al. [96] propose an automated context-aware anomaly assessment rule-set framework based on vulnerability signatures. This method uses a cumulative sum of residuals on historical system data to detect stealthily changing variables. It considers noise patterns in sensors (e.g., sensor accuracy level or water movement in a tank) as context information for this proposal.

In recent years, the work presented in [43] developed control charts using machine learning for monitoring communication between nodes modeled by a Poisson distribution. Chang et al. [44] generated three spectral and spatial maps from an anomaly map: ADMap, spectral FGMap, and spatial SFMap. These maps provide feedback information. Also, [45] proposed an algorithm for detecting anomalies in real-time data with mixed attributes. The algorithm combines both a partitioning and an agglomerative hierarchical approach. It produces clusters, each associated with a fuzzy time interval, representing the duration of its existence.

3.2.2 Machine Learning

We identified several machine learning methods used to detect anomalies, such as support vector machine [2, 104–107], k-nearest neighbors [108, 109], decision trees [105, 110, 111], optimized gradient boosting decision tree [23, 112, 113], isolation forest [114, 115], and spatial density-based clustering of applications with noise [116].

Table 3.2: Anomaly detection proposals using machine learning methods.

Reference	**Anomaly detection strategy**	**Anomaly**	**Classify (E-F-A)**	**Context Aware**	**Detection Layer**	**Execute Layer**
(Li, 2024) [2]	Random Forest and KNN	ALL	✓	✗	Network	-
(Niu, 2023) [23]	Random Forest and gradient boosting	Attacks	✓	✗	Network	-
(Ahakonye, 2023) [117]	Decision tree and Chi-square for feature selection	Attacks	✓*	✗	Network	-
(Douiba, 2023) [113]	Decision tree and gradient boosting	Attacks	✓*	✗	Network	Local server
(Yang, 2022) [114]	Detect data distribution change in time and train the new model	All	✗	✗	Perception	-
(Rousopoulou, 2022) [104]	Generic platform for anomaly detection	Failures	✗	✗	Perception	Cloud
(Su, 2022) [118]	Machine-learning tree-based methods	Attacks	✗	✗	Network	-
(Rey, 2022) [119]	Autoencoder in federated learning	Attacks	✗	✗	Perception	Edge
(Kumar, 2022) [105]	Botnet detection using network-edge traffic	Attacks	✗	✗	Network	Edge
(Garmaroodi, 2020) [106]	Data mining	Failures	✗	✗	Perception	Edge
(Cui, 2021) [112]	Margin synthetic minority oversampling technique for unbalanced data	Attacks	✗	✗	Network	Edge
(Elnour, 2021) [115]	data-driven attack detection using Isolation Forest	Attacks	✗	✗	Perception	-
(Yang, 2020) [108]	Secure vector homomorphic encryption scheme	All	✗	✗	Perception	Cloud
(Razzak, 2020) [107]	Randomized nonlinear one-class support vector machine	All	✗	✗	Perception	-
(Bodo, 2020) [110]	Feature selection method based on hierarchical feature ranking	All	✗	✓	Perception	-
(He, 2020) [120]	Decision triggered data transmission and collection protocol	All	✗	✗	Perception	-
(Garg, 2020) [116]	Density-Based Spatial Clustering of Applications with Noise (DBSCAN)	Attacks	✗	✗	Network	-
(Zhang, 2020) [121]	Maximum correlation minimum redundancy feature selection algorithm	Attacks	✗	✗	Network	-
(Anton, 2019) [122]	Matrix Profiles detect attacks that occur multiple times	Attacks	✗	✗	Network	-
(Raposo, 2019) [123]	Use on-node metrics available in hardware	All	✗	✓	Perception	-
(Raposo, 2018) [124]	One Class Support Vector Machine	Attacks	✗	✓	Network	Edge
(Shi, 2019) [109]	Extract statistical and spectral features	Attacks	✗	✓	Network	-
(Ouyang, 2018) [125]	Multi-view learning based ensemble learning solution	Events	✗	✗	Perception	Cloud

Methods marked with * classify anomalies without identifying the three kinds of anomalies defined in this review.

Table 3.2 summarizes information extracted from the papers using machine learning techniques. This table identifies several proposals that detect anomalies of different natures without diagnosing the origin. Authors in [123] use metrics from a specific microcontroller brand to detect firmware and hardware anomalies (e.g., buffer overflow attacks, SPI failures, voltage drops, and high-temperature failures). However, it assumes that attacks cause all anomalies. In this case, metrics delivered by the microcontroller unit (e.g., execution time, energy counter, microcontroller unit cycles) could be considered context information. Bodo et al. [110] used decision trees to determine whether data correspond to an anomaly. Although they do not determine the origin of an anomaly, this strategy could detect whether data are labeled appropriately. The authors use specifications from detection devices, environmental noise, and available processing resources to support detecting an anomaly, which could be considered context information.

Another work presented in [109] uses the power consumption of an IoT device to detect anomalies. In this case, a device's power consumption could be considered context information. The report in [120], in turn, identifies anomalies by comparing suspicious data from a specific sensor against data from other sensors recording similar variables. However, they do not determine whether it is a failure or an attack in the application context. Moreover, authors in [113, 117] classify the specific type of attack using decision trees. Niu et al. [23] proposed ADESSA, a model combining semi-supervised learning with active learning. It manually labels *difficult-to-judge* unlabeled samples, which contain more new information than other samples. ADESSA also uses active learning to discover unknown attacks in unlabeled samples, enabling CPS managers to capture novel attacks. The base classifier used in this work was random forests and gradient-boosted trees. In contrast, [2] trained decision trees, Random Forest, k-nearest Neighbors, Naive Bayes, and Multiple Layer Perception models for binary and multiple classification to determine the effectiveness of two feature reduction methods.

3.2.3 Deep Learning

Neural networks are used in most papers for anomaly detection. The most common models used are Convolutional Neural Networks (CNN) [126–128], Recurrent Neural Networks (RNN) [129, 130], bidirectional long and short-term memory (LSTM) [131–134], variational autoencoders [135–138], CNN-LSTM [139], [140], [141], and Transformers [12, 24, 37, 105, 142, 143].

Table 3.3 presents papers proposing anomaly detection using Deep Learning techniques. Some works use neural networks as multilabel classifier [33, 90, 129, 141, 144–146]. For example, authors in [33] combine one-dimensional convolution neural networks and the Dempster–Shafer decision fusion method to detect and classify some specific failure types, and authors in [146] use a robust multi-cascaded convolutional neural networks (CNN) classification approach to distinguish between Sybil and DoS attacks. The deep neural network architecture developed by [141] incorporates inherent convolutional neural networks, which act as a multi-label classifier to determine the intrusion points of attacks. The work presented in [147] trains a multi-class fault classification auto-encoder using sensor measurements collected during faulty operation.

Table 3.3: Anomaly detection proposals using deep learning methods.

Reference	Anomaly detection strategy	Anomaly	Classify (E-F-A)	Context Aware	Detection Layer	Execute Layer
(Priyadarsini,2025) [148]	CNN to track the injected attack types	Attacks	✗	✗	Network	Edge
(Saheed,2024) [149]	Genetic algorithm with attention mechanism	All	✗	✗	Perception	-
(He,2024) [150]	Deep-learning side-channel analysis	Attacks	✗	✗	Network	Edge
(Abudurexiti,2024) [151]	Kolmogorov–Arnold Network	Attacks	✗	✗	Perception	-
(Fan,2024) [152]	Spatio-Temporal Gated Attention Networks	All	✗	✗	Perception	-
(Bukhari,2024) [153]	Asynchronous federated learning	Attacks	✗	✗	Perception	-
(Chang,2024) [154]	Generative Adversarial Network	Failures	✗	✗	Network	-
(Yu,2024) [155]	GRU encoders	Attacks	✗	✗	Network	-
(Sankaran,2023) [146]	Multi-cascaded CNN classification	Attacks	✓*	✗	Network	-
(Cavdar,2023) [33]	1D convolution neural networks and the Dempster–Shafer	Failures	✓*	✗	Perception	-
(Halder,2023) [156]	Federated learning with GRU	Attacks	✗	✗	Network	Local server
(Kumar,2023) [157]	An adaptive transformer model for anomaly detection	Attacks	✗	✗	Network	-
(Kim,2023) [37]	Stacked Transformer representations and 1D Convolutional network	Failures	✗	✗	Application	-
(Ba, 2022) [24]	Automated Configuration of Heterogeneous Graph Neural Networks	All	✗	✗	Perception	-
(Ba,2022) [158]	Transformer-based Graph Convolutional Neural Networks	All	✗	✗	Network	-
(Truong,2022) [12]	Light-weight federated learning-based anomaly detection	All	✗	✗	Perception	Edge
(Hu,2022) [159]	Transfer Learning based Trajectory Anomaly Detection	All	✗	✗	Perception	Edge
(Pan,2022) [160]	Dual masked self-attention mechanism	All	✗	✗	Perception	-
(Feng,2022) [161]	A full graph autoencoder	All	✗	✗	Perception	-
(Nizam,2022) [9]	Convolutional neural network and a two-stage LSTM based Autoencoder	All	✗	✗	Perception	-
(Mukherjee, 2022) [141]	Deep learning models to determine the exact intrusion points in real-time	Attacks	✗	✗	Perception	-
(Nedeljkovic, 2022) [126]	Method for calculating the hyperparameters of CNN to detect cyber-attacks	Attacks	✗	✗	Network	Device
(Weinger, 2022) [162]	Data augmentation in federated learning for anomaly detection	All	✗	✗	Perception	Edge
(Friha, 2022) [163]	Federated learning-based decentralized intrusion detection system	Attacks	✗	✗	Network	Edge
(Liu, 2022) [127]	DDoS detection with information entropy analysis	Attacks	✗	✗	Network	-
(Yang, 2022) [164]	One-class broad learning system	Attacks	✗	✗	Network	-
(Chen,2021) [142]	Learning Graph Structures With Transformer	Attacks	✗	✗	Perception	-
(Wangwang,2021) [165]	Network Traffic Oriented Malware Detection	Attacks	✗	✗	Network	-
(Kozik,2021) [143]	A hybrid time window embedding with transformer-based traffic data classification	Attacks	✗	✗	Network	-
(Wang,2021) [166]	Hierarchical Federated Learning	Attacks	✗	✗	Perception	-
(Wang, 2021) [131]	Unknown attack Identification using spatial-temporal features	Attacks	✗	✗	Network	-
(Huong, 2021) [135]	VAE-LSTM model on edge devices	Attacks	✗	✗	Perception	Edge
(Kong, 2021) [132]	Generative adversarial networks	All	✗	✗	Network	-
(Wang, 2021) [166]	Federated deep reinforcement Learning	Attacks	✗	✗	Network	-
(Khan, 2021) [140]	Temporal and spatial features for the classification and explanation attacks	Attacks	✗	✗	Network	-
(Ketonen, 2021) [167]	Probabilistic Deep Learning	All	✗	✗	Perception	-
(Liu, 2020) [168]	Attention Mechanism-Based CNN Unit and LSTM Unit	All	✗	✗	Perception	Edge
(Savic, 2021) [136]	Autoencoder in edge device	All	✗	✗	Perception	Edge
(Seo, 2021) [128]	Acoustic-Based Anomaly Detection	Attacks	✗	✗	Perception	Edge
(Dzaferagic, 2021) [147]	Generative Adversarial Networks to generate missing sensor measurements	Failures	✗	✗	Perception	-
(Wu, 2021) [169]	Graph Neural Networks	All	✗	✗	Perception	Edge & Cloud
(Zhou, 2020) [170]	LSTM to mitigate dimensional reduction in unbalanced data	All	✗	✗	Network	-
(Li, 2020) [171]	multi-CNN fusion	Attacks	✗	✗	Network	-
(Park, 2020) [129]	Setting boundaries based on cosine similarity in network packets	All	✗	✗	Network	-
(Wu, 2019) [133]	LSTM with Bayesian and Gaussian Processing	All	✗	✗	Perception	-
(Li, 2020) [134]	Bidirectional long and short-term memory (B-LSTM)	Attacks	✗	✗	Network	Local server
(Liu, 2020) [168]	On-device collaborative deep anomaly detection	Failures	✗	✗	Perception	Edge
(Demertzis, 2020) [172]	Blockchained deep learning smart contracts	All	✗	✓	Application	Cloud
(De Vita, 2020) [173]	DeepAutoencoder and PCA blocks	Failures	✗	✗	Perception	Edge & Cloud
(Liu, 2020) [139]	Federated Learning to collaboratively train a Deep Anomaly Detection	All	✗	✗	Perception	Edge
(Wang, 2020) [130]	Recurrent neural networks	Failures	✗	✗	Perception	Edge
(Ferrari, 2019) [174]	Compare LSTM on edge and cloud	Failures	✗	✗	Perception	Edge & Cloud
(Liu, 2019) [175]	Gated Recurrent Unit (GRU) and Support Vector Domain Description	Attacks	✗	✗	Network	-
(Bernieri, 2019) [137]	Variational Autoencoders(VAE)	Attacks	✗	✗	Network	-
(Al-Hawawreh, 2019) [138]	Variational Auto-Encoder learns the latent structure of system activities	Attacks	✗	✗	Network	-
(Krundyshev, 2019) [176]	Determining the normal (legitimate) activity of nodes	Attacks	✗	✗	Network	-
(Bae, 2018) [177]	Autoencoder with invasion scoring	Attacks	✗	✗	Network	-
(Al-Hawawreh, 2019) [145]	Sparse and denoising autoencoder	All	✗	✗	Network	Local server
(Kim, 2018) [178]	Squeezed Convolutional Variational AutoEncoder	All	✗	✗	Perception	Edge
(Saurav, 2018) [36]	Recurrent Neural Networks Recurrent Units (GRU)	All	✗	✗	Network	-
(Muna, 2018) [179]	Deep Auto-Encoder (DAE)	Attacks	✗	✗	Network	Cloud
(Schneider, 2018) [180]	Stacked denoising autoencoder	Attacks	✗	✗	Network	Edge
(Tandiya, 2018) [181]	Frequency-domain data are transformed in 2D image	Attacks	✗	✗	Network	Edge

Methods marked with * classify anomalies without identifying the three kinds of anomalies defined in this review.

The work presented in [130] uses RNN to detect anomalies and provides insights into the timestep at which an anomaly occurred. This system assists a human operator, which, in turn, locates the source of a problem. Whereas authors in [33, 146] classify the specific type of attack or failure, respectively. Regarding neural network models, some works use long short-term memory (LSTM) to leverage spatial and temporal correlation on anomaly detection [133, 134, 174]. Some results use Principal Components Analysis (PCA) to reduce the dimensions in a dataset before training a neural network [173, 175]. Other papers use auto-encoder networks and only train models with normal operating data, which avoids dealing with rare anomalous data in an industrial system [178–180].

Working with big data in real-time anomaly detection systems presents several challenges because of the constrained resources of memory and processing power of edge devices and the high latency of cloud computing processing. A solution to take advantage of IIoT characteristics is the federated learning technique, which is used for training and detecting anomalies in a distributed way [166, 168]. In the last years, modified versions of Transformer and the attention mechanism [49] have gained momentum in the field of anomaly detection [12, 24, 37, 105, 142, 143], some authors use graph-CNN for feature selection and transformers for anomaly detection [24, 142, 158], other works use auto-encoders based on transformers for the same task [12, 98]. In addition, transfer learning with a variational graph autoencoder is used in a trajectory anomaly detection strategy [159]. Proposals in this section neither perform automatic classification of the anomaly origin as an event, attack, or failure nor identify information that could be considered context information.

Zhang et al. [10] designed an algorithm for detecting anomalies in multivariate time series. It utilizes a Transformer Block embedded into a Variational Auto Encoder architecture. While [37] presents an unsupervised methodology for time-series anomaly detection utilizing multiple Transformer encoder layers and a decoder with a 1D convolution layer. Ding et al. [13] developed a new approach to enable the Transformer model to adapt to concept drift by adjusting its learning rate dynamically. This technique is beneficial because the standard Transformer model assumes that the training and test data come from the same distribution, which is frequently not the case in real-world scenarios. This is especially significant for time-series data, where the data characteristics can change over time, leading to concept drift problems.

Truong et al. [12] introduced a decentralized architecture for detecting anomalies. The architecture combines Federated Learning, Autoencoder, Transformer, and Fourier mixing sublayer techniques. It is lightweight and requires minimal CPU and memory resources. It also consumes low bandwidth, making it possible to deploy it on edge devices with limited computing capabilities. In addition, [11] presents TranAD, a transformer-based model for detecting and diagnosing anomalies. The model uses attention-based sequence encoders to enable fast inference and account for broader temporal trends in data.

In its work, Zeng et al. [50] proposed Adformer, a model designed to detect anomalies in multi-dimensional time series data in IoT applications. The model uses two-stage adversarial training to improve the transformer's ability to capture short-term trends in the time series and amplify the reconstruction error. This makes it easier to distinguish between normal and abnormal points with greater accuracy. In the research [55], authors present a solution for detecting anomalies called the Ensemble Denoising Auto-Encoder-based Dynamic Threshold (EDAE-DT). This approach aims to address the issue of false alarms by utilizing a denoising task and an ensemble approach. By modeling normal behavior accurately, the technique sets a time-varying threshold that considers the variation of normal data.

Jithish et al. [46] describe a smart grid anomaly detection system that uses Federated Learning (FL) technology. In this model, machine learning models are trained locally in smart meters. This is done without sharing the data with a central server, ensuring user privacy. The system downloads a global model from the server to the smart meters for on-device training. After local training, the local model parameters are sent to the server to improve the global model. Alternatively, Saez et al. [47] present a federated learning architecture for training anomaly detectors in large networks of heterogeneous IoT devices. The system uses the FL framework to train the anomaly detection models collaboratively between multiple participants without sending each device's local data. This approach reduces network overhead and addresses data isolation and privacy concerns.

Autoencoders are unsupervised models commonly used for anomaly detection. They identify anomalies in the original data by analyzing the differences between the input and output data [53]. In [33], the Dempster–Shafer decision-fusion method is combined with one-dimensional convolution neural networks for anomaly decision-making in IIoT. While [53] proposed an LSTM-based autoencoder with several decoders. Each decoder handles data from a specific flight phase, such as climbing, cruising, or descending. In the same way, [48] proposed a Dilated Convolutional Transformer-based GAN (DCT-GAN) to improve model accuracy and generalization. Multiple Transformer-based generators were designed and integrated using a weight-based mechanism to ensure compatibility with various anomalies.

Since the autoencoders are trained with normal data, it is necessary to calculate a threshold value to detect anomalies. Authors in [55] propose determining a threshold using residuals. Residuals are obtained by calculating the L1 norm or L2 norm of the difference between the output and true data. Specifically, if the confidence level is p, the threshold can be set as the value at which the cumulative distribution function of a residual becomes (1-p). Ultimately, the system's condition is monitored by comparing the residual with the threshold.

3.3 Anomaly Classification

Identifying anomalies is extremely important for industrial systems as it allows operators to respond appropriately to attacks and failures [182]. It is crucial to distinguish between anomaly types to take the appropriate action for component reconfiguration, estimate the extent of propagation in the system, and prevent negative reactions that could worsen the system's condition. In industrial systems, it is decisive to minimize response times to address anomalies in critical infrastructure due to strict real-time requirements. Therefore, quick and accurate detection and classification of anomalies are vital [16, 26]. In the state of the art, very little research classifies anomalies in IIoT; most research classifies anomalies in the industrial process and cyber-physical systems (CPS). Some of those works are described below,

FALCON is a system that detects failures in power distribution systems by identifying short-circuit faults through significant variations in current and voltage. It can detect false-data injection and replay attacks [17]. The authors used the IEEE 13 bus feeder simulator to test the system and a multilayer perceptron as a classifier with optimized hyperparameters.

Yang et al. [15] proposed a data-driven approach for identifying and distinguishing between physical faults and cyber-attacks in manufacturing motor drives. The method involved a simulated dual-motor-drive network within manufacturing systems. It utilized line current spectra and four data-driven classifiers to detect cyber-attacks, such as data injection, DoS, eavesdropping, and replay, and physical faults, such as inter-turn short circuits and bearing faults. The method employed four classifiers, namely, KNN, SVM, RF, and LR, which conducted a majority vote to generate an alarm if three or more classifiers detected a fault or an attack.

In its work, Zhang et al. [16] described a methodology that aims to distinguish between two types of threats in Cyber-Physical Systems: sensor replay attacks and sensor bias faults. To achieve this, the method integrates a watermark with adaptive estimation. A distinct watermark is created for each threat type based on the changes it causes in the system. The authors validate their approach using a simulation. Liang et al. [18] address the security challenges of multi-agent systems (MAS) due to physical faults and cyberattacks. An observer-based distributed attack detection and fault diagnosis scheme is proposed to minimize communication loss. Each agent has a bank of attack detectors based on an unknown input observer to detect attacks in communication with other agents. A fault diagnoser is also designed for each agent, which can be combined with the attack detectors to make a distributed fault diagnosis. The proposed scheme is tested on two smart grids with multiple generators to verify its effectiveness and scalability.

SCADA systems manage and control critical infrastructures such as power plants and manufacturing systems. In [134], four types of equipment fault scenarios are derived from statistical measures to detect each type of fault and differentiate it from a replay attack. They created a data-driven methodology using a Kalman filter and linear-quadratic Gaussian controller and conducted physical experiments on a rotating machinery testbed to examine its practicality. [183] proposes a methodology to detect and isolate simultaneous cyberattacks and faults in interconnected cyber-physical systems. It uses two filters on the plant and control sides of the CPS, along with an unknown input observer-based detector on the plant side. The methodologies can identify various attacks, including covert, zero dynamics, and replay attacks. Simulations using the OPAL-RT real-time simulator and 4 Raspberry Pi demonstrate the capabilities of the methodology on a four-area power network system.

Abu et al. [184] suggest using convolutional neural networks running in a Compute Unified Device Architecture (CUDA) based on Nvidia GPUs (Graphical Processing Units) to detect and classify cyber-attacks in IoT communication networks NSL-KDD dataset. Similarly, the authors in [185] described four machine-learning-based decision tree models, namely AdaBoosted, RUSBoosted, bagged, and their ensemble learning model for classifying botnet attacks in the NBaIoT2021 dataset. In another study, [186] utilized AdaBoost machine learning algorithms combined with Decision Trees to classify some attacks in an IIoT dataset, such as DoS, DDoS, MitM, backdoor, and injection. In addition, [187] analyzes how feature selection can increase detection accuracy and speed up the training phase. They test their proposed classification approach on Mirai, DoS, Scan, MAS (MitM-ARP Spoofing) attacks, and normal operation.

Authors in [26] propose classifying sensor anomalies into four categories: stealthy attack, random attack, transient failure, and permanent failure. To do this, it detects risks, analyzes them, and reconfigures the system if a group of damages exceeds a tolerance threshold. However, the solution requires a deep knowledge of the system. The authors employ machine learning techniques to address this issue stem to detect physical failures and cyber-attacks in critical infrastructure using a testbed that simulates a water plant. They investigate how the monitoring modules react to physical and cyber problems while analyzing cross-effects. Some proposals distinguish between two types of threats in Cyber-Physical Systems: physical failures and cyberattacks. The authors in [188] aimed to distinguish between component failures and attacks on a power-aware communication network in an intelligent home system. They provided a framework and suggested that a normal state is when all variables are within expected limits with no failures and attacks. However, if the effects of failures and attacks are similar, distinguishing between the two is impossible with the provided framework.

In their work, Atul et al. [22] propose an Energy Aware Smart Home (EASH) framework. The study investigates the types of network attacks and communication failures that occur in EASH. The authors employ machine learning techniques to address this issue to distinguish abnormal communication patterns. The authors classify normal communication patterns and anomalies (failures and attacks). In [32], authors try to differentiate between two types of anomalies in an MIoT network - Presence-Hard-Contact Anomalies and Success-Hard-Content Anomalies, based on the number of transactions. Two approaches are used - forward and inverse problems. The forward problem aims to comprehend the impact of anomalies on the nodes. The inverse problem is used to detect the origin of anomalies by studying their impact on the nodes.

Table 3.4 from our review article [14] presents the papers that detect general anomalies or events, failures, and attacks. Attack detection receives the most attention from researchers in this field, followed by the detection of general anomalies.

Most works on detecting anomalies systems prioritize malicious attack detection, with a smaller percentage dedicated to detecting faults. It is essential to analyze and distinguish the source of anomalies to select the appropriate recovery actions. Automatic diagnostic functions are necessary to eliminate the need for trained operators [188].

Table 3.4: Type of detected anomaly.

Events	Failures	Attacks
(Peng,2019) [102], (Ouyang,2018) [125]	(Ferrari,2019) [174], (De Vita,2020) [100], (Liu,2020) [168], (De Vita,2020) [173], (Wang,2020) [130], (De Vita,2021) [83], (Garmaroodi,2020) [106], (Rousopoulou,2022) [104], (Dzaferagic,2021) [147], (Kim,2023) [37], (Cavdar,2023) [33]	(Garitano,2019) [87], (Anton,2019) [122], (Raposo,2018) [124], (Li,2020) [171], (Liu,2019) [175], (Bernieri,2019) [137], (Wang,2020) [101], (Al-Hawawreh,2019) [138], (Bernieri,2019) [91], (Krundyshev,2019) [176], (Shi,2019) [109], (Li,2020) [134], (Bae,2018) [177], (Bernieri,2019) [92], (Gorbenko,2020) [84], (Garg,2020) [116], (Enuachescu,2019) [93], (Aoudi,2020) [85], (Tandiya,2018) [181], (Wang,2021) [131], (Huong,2021) [135], (Zhang,2020) [121], (Cui,2021) [112], (Wang,2021) [166], (Hashmat,2022) [96], (Khan,2021) [140], (Mukherjee,2022) [141], (Elnour,2021) [115], (Cai,2021) [81], (Nedeljkovic,2022) [126], (Seo,2021) [128], (Weinger,2022) [162], (Su,2022) [118], (Rey,2022) [119], (Friha,2022) [163], (Kumar,2022) [105] (Liu,2022) [127], (Yang,2022) [164], (Ghaeini,2018) [94], (Zugasti,2018) [95], (Muna,2018) [179], (Schneider,2018) [180], (Madhawa2018) [103], (Chen,2021) [142], (Wangwang,2021) [165], (Kozik,2021) [143], (Kumar,2023) [157], (Wang,2021) [182], (Douiba,2023) [113], (Halder,2023) [156], (Ahakonye,2023) [117]
General Anomaly		
(Raposo,2019) [123], (Yang,2020) [108], (Park,2020) [129], (Razzak,2020) [107], (Faisal,2019) [88], (Wu,2019) [133], (Bodo,2020) [110], (He,2020) [120], (Demertzis,2020) [172], (Genge,2019) [89], (Al-Hawawreh,2019) [145], (Li,2020) [86], (liu2020) [139], (Kong,2021) [132], (Zhan,2021) [99], (Aruquipa,2022) [97], (Ketonen,2021) [167], (Yang,2022) [114], (Liu,2020) [168], (Savic,2021) [136], (Wu,2021) [169], (Dang,2021) [82], (Wang,2022) [98], (Zhou,2020) [170], (Kim,2018) [178], (Saurav,2018) [36], (Ba,2022) [24], (Ba,2022) [158], (Truong,2022) [12], (Wang,2022) [98], (Hu,2022) [159], (Pan,2022) [160], (Sankaran,2023) [146], (Feng,2022) [161], (Nizam,2022) [9]		

3.4 Context Information

Context is a term used to describe any valuable information that provides a fuller understanding of an entity's situation. This entity, a machine, location, or object, plays an important role in the interaction between a user and an application or among applications. By providing context, the system can better understand the situation and make informed decisions [63]. This section will reference some works that use information that could be considered contextual, although the authors do not use that term.

The paper [24] proposes a method that automatically extracts mathematical relationships and potential features from documents comprising mathematical formulas and natural language. The derived features are then added to the dataset. The paper also describes how the extracted information and specific domain knowledge are combined to create a semantic knowledge graph for modeling purposes. They do not use the term "context information" but "domain knowledge" and use those derived features directly in the machine learning model. In [189], the authors propose a new method for cleaning environmental sensor data that considers contextual information from external sources. This approach includes integrating geographical and meteorological data into the training dataset for building statistical models and using this data during the online cleaning of the environmental data stream. The contextual data is obtained through web services [189]. The authors of [24] have employed a Transformer as a learner that operates from sequence to sequence, allowing the direct integration of domain knowledge into ML models. This integrates datasets and derived features and aims to scale ML to IoT applications by modeling the relationship between the extracted features and response variables.

MOM (Modeling, Organization, Middleware) is proposed by [63]. Context-aware systems have three primary components: generic and effective context modeling, efficient context organization, and robust context middleware. The significance of context is crucial in imbuing information with intelligence. This is because the background, circumstances, and conditions in which information is presented play a fundamental role in understanding and interpreting its meaning. Garitano et al. [87] detect attacks, sensor failures, and communication problems due to a man-in-the-middle attack. It examines the contribution of each variable to an abnormal event. However, the proposal does not automatically identify the source of the anomaly. This limits the functionality of generating an alarm for a human operator. The human operator must then diagnose whether it is a plant event, a communication event, or an attack. The authors use "metadata", which incorporates physical and network variables. This data can be considered context information.

3.5 Discussion

The Industrial Internet of Things (IIoT) transforms industries through improved connectivity, data gathering, and real-time monitoring capabilities. However, the growing use of connected devices and sensors makes these systems vulnerable to various irregularities and security risks. This chapter has thoroughly examined the progress made in anomaly detection and classification techniques in the IIoT field, emphasizing the necessity for reliable systems that can effectively differentiate between normal and abnormal behaviors.

Our review categorizes anomaly detection techniques into three main groups: statistical methods, traditional machine learning, and advanced deep learning algorithms. Each group offers unique strengths and challenges. Statistical techniques use mathematical models to identify deviations from normal behavior. Although effective for detecting simple anomalies, they often struggle with complex or sophisticated threats. Control charts, wavelet transforms, and Bayesian models are common in this category. For instance, the work of Garitano et al. (2019) uses metadata to detect sensor failures and communication issues, demonstrating the potential of statistical methods to incorporate contextual information. While machine learning approaches, such as support vector machines, k-nearest neighbors, and decision trees, have shown promise in detecting anomalies by learning from historical data. These methods are more adaptable to varying conditions than statistical methods. For example, Raposo et al. [123] use metrics from microcontroller units to detect firmware and hardware anomalies, leveraging contextual data to enhance detection accuracy. In addition, deep learning algorithms, including convolutional neural networks, recurrent neural networks, and autoencoders, offer advanced capabilities for handling large and complex datasets. These methods are particularly effective in detecting sophisticated cyber-attacks and system failures. Recent advancements, such as using Transformers and federated learning, highlight the trend towards decentralized and context-aware anomaly detection. For instance, Truong et al. [12] introduce a lightweight federated learning-based approach for edge devices, reflecting the push towards efficient and scalable solutions.

Our analysis reveals several key trends and challenges in IIoT anomaly detection. First, there is a clear shift towards leveraging deep learning techniques for anomaly detection, particularly on edge devices. This trend aligns with the growing need for real-time and on-device processing capabilities in IIoT systems. Second, incorporating contextual information into anomaly detection algorithms significantly enhances their accuracy and reliability. However, only a small percentage (8%) of the reviewed papers utilize context-aware information, indicating a gap in current research. Third, nearly half (48.9%) of the reviewed papers focus on deploying anomaly detection systems on edge devices. This reflects the industry's move towards edge computing to reduce latency and improve real-time decision-making. Finally, a significant number of the papers only identify anomalies without classifying their origin, emphasizing the necessity for research that establishes the groundwork for systems capable of automatically managing various anomalies.

Based on our review, we identified several promising directions for future research in IIoT anomaly detection. Future research should focus on developing more sophisticated methods for incorporating contextual information into anomaly detection algorithms. This could involve integrating data from various sources, such as environmental sensors, operational logs, and network traffic. Likewise, with the growing adoption of edge and fog computing, there is a need for scalable and decentralized anomaly detection systems, like federated learning, but further exploration is required to optimize their performance and security. Moreover, the application of advanced deep learning models, such as Transformers and graph neural networks, should be further explored to handle the complexity and scale of IIoT data. These models can potentially improve the detection of subtle and sophisticated anomalies. Additionally, there is a need for standardized benchmarks and datasets to evaluate and compare the performance of anomaly classification methods. This would facilitate more rigorous and reproducible research, enabling the identification of best practices and gaps.

In conclusion, the advancements in anomaly detection and classification within the IIoT landscape ensure the security and reliability of industrial systems. While significant progress has been made, particularly with adopting deep learning and edge computing, challenges remain in enhancing context-awareness and scalability. The insights and contributions presented in this chapter underscore the importance of developing robust and intelligent anomaly detection systems that can adapt to the evolving threats and complexities of the IIoT environment. This chapter has provided a structured overview of current methodologies for IIoT anomaly classification. It has inspired the work described in this thesis, where a framework is proposed to classify anomalies in IIoT, executing a model based on deep learning that incorporates context information on edge devices.

Chapter 4

Framework for Anomaly Classification

The rapid evolution of the Industrial Internet of Things (IIoT) and its integration with Cyber-Physical Systems (CPS) have paved the way for the emergence of smart factories [20]. These advanced manufacturing environments leverage digital technologies to enhance operational efficiency, productivity, and flexibility [10, 12, 23]. However, these systems' complexity and interconnected nature introduce significant challenges in monitoring and securing industrial processes. Anomalies, from equipment failures to cyber-attacks, can disrupt operations, cause financial losses, and threaten safety [6–8, 27]. Thus, there is a critical need for sophisticated anomaly detection and classification mechanisms tailored to the unique requirements of the IIoT ecosystem within smart factories. This ensures the smooth operation of manufacturing processes and the safety and security of the underlying industrial infrastructure.

In response to these challenges, this comprehensive framework delineates a novel approach to anomaly detection and classification designed for the IIoT landscape. By integrating specific domain knowledge of the CPS with the data-driven insights provided by IIoT technologies, the framework aims to enhance the accuracy and efficiency of anomaly classification. It meticulously outlines methodologies for context feature analysis, data collection, feature selection, and data processing, culminating in advanced strategies for balancing data and optimizing anomaly detection models. Emphasizing the importance of understanding the IIoT architecture and the operational nuances of the industrial processes it monitors, this framework represents a significant step forward in operationalizing AI and machine learning technologies for real-world industrial applications, thus clearing the way for more resilient, intelligent, and self-optimizing smart factories. This chapter presents a comprehensive framework for detecting and classifying anomalies in Industrial Internet of Things (IIoT). Here are the key original contributions of this chapter:

- Unique Focus on Anomaly Classification within IIoT: Unlike broader research that might cover fault diagnosis across industrial machines, this framework specifically targets the classification of anomalies within the IIoT system. It emphasizes the necessity of understanding the IIoT architecture and the underlying industrial processes, advocating for a deep integration of specific domain knowledge directly into machine learning models for IIoT applications.

- Comprehensive Methodology for Context Feature Analysis and Data Collection: The chapter introduces a methodology for identifying and incorporating context features and variables that can aid anomaly classification. It also discusses various data collection strategies, including using public repositories, simulators, historical data, and testbeds. It highlights the complexity and challenges of gathering adequate training data for anomaly detection models.

- Techniques for Feature Selection and Data Processing: The framework examines feature selection techniques, including exploratory data analysis and dimensionality reduction, to enhance model accuracy and efficiency. It also covers the essential steps of data processing, such as data integration, handling missing values, removing duplicates, and encoding non-numerical features, ensuring high-quality data preparation for machine learning applications.

- Strategies for Data Balancing and Anomaly Detection Optimization: Addressing the common issue of imbalanced datasets in anomaly detection, the framework presents strategies for data augmentation and resampling methods, including the application of Synthetic Minority Over-sampling Technique (SMOTE) and Generative Adversarial Networks (GANs). It also discusses the optimization of anomaly detection models through careful selection of evaluation metrics and hyperparameter optimization techniques.

- Two-stage Approach to Anomaly Classification: A contribution of the framework is its two-stage approach to IIoT anomaly classification, which first identifies data as normal or anomalous before classifying the type of anomaly (failure or attack). This approach enables a more nuanced understanding and response to anomalies detected in the IIoT system.

- Detailed Guidance on Evaluation Metrics and Model Optimization: The framework offers in-depth guidance on selecting appropriate evaluation metrics based on the classification task and dataset characteristics. It also provides insights into the optimization process, emphasizing the importance of tuning model parameters and selecting algorithms to improve classification accuracy and reduce false positives and negatives.

Overall, this chapter's original contributions lie in its detailed and systematic approach to anomaly detection and classification within the IIoT context. It offers innovative strategies for data collection, feature selection, model optimization, and classification that can significantly improve the security and efficiency of industrial operations. Figure 4.1 shows the outline of the proposed framework to classify anomalies in IIoT.

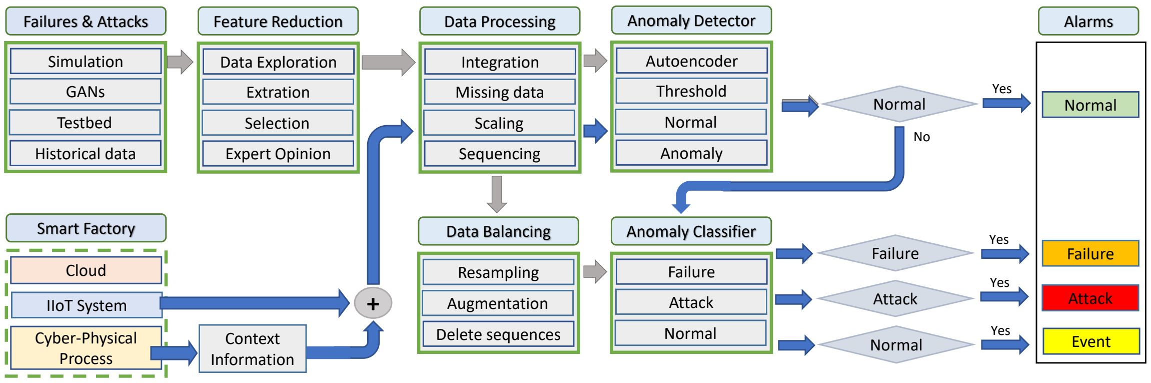

Figure 4.1: Framework for Anomaly Classification in IIoT. During training (grey arrows), data is collected from a testbed or dataset or is artificially generated. Features are then selected, preprocessed, and used to train the anomaly detector, and classes are balanced before training the classifier. In deployment (blue arrows), data from the CPS and the IIoT system is processed before feeding the detector. If a sample is labeled as an anomaly, the classifier is invoked to classify it as a failure or attack. But if there is ambiguity between the detector and the classifier, an "event" alert is generated for a human operator to classify the sample.

This chapter explores the proposed framework for anomaly classification in Industrial Internet of Things. We introduce the smart factory concept, highlighting the integration of Cyber-Physical Systems (CPS) with IIoT technologies and emphasizing the necessity of understanding these systems in detail. The chapter then discusses the core components of the framework, starting with the CPS and IIoT architecture and explaining their roles and interactions. Detailed data collection, processing, and feature selection steps are provided to ensure a robust and comprehensive dataset. The chapter also includes sections on balancing data, anomaly detection techniques, and classification strategies, emphasizing the use of advanced machine learning models. Finally, we discuss the evaluation metrics and optimization techniques used to enhance model performance. Each section is designed to equip practitioners and researchers with the necessary knowledge and tools to effectively implement and deploy an anomaly classification system in IIoT settings.

4.1 Smart Factory

In this context, the term *smart factory* refers to the integration of the cyber-physical system (CPS) with the Industrial Internet of Things (IIoT) system. The CPS is an automated industrial process that uses SCADA and programmable logic controllers (PLC). CPS is the core component of the business [10, 12, 23]. The IIoT system is installed to monitor the CPS through a cloud platform, optimizing its performance. In this framework, the first activity is to know the CPS and IIoT that already operate in the plant.

4.1.1 Cyber-Physics System

A Cyber-Physical System (CPS) is characterized by the integration between physical processes and computational (cyber) processes facilitated through the Industrial Internet of Things (IIoT) technologies [22]. Several research and business solutions propose fault diagnosis directly in industrial machines. However, this framework offers a way to classify anomalies in the IIoT system, not the industrial process.

This framework assumes that technical personnel designing the anomaly classifier for IIoT know how the cyber-physical system (industrial process) works. They possess expertise or a deep understanding of a particular area [24]. This domain knowledge includes the concepts, techniques, tools, and vocabulary unique to that domain [25]. For instance, the maintenance team must know how the industrial process works and the expected ranges of variables and recognize what is normal. Although this framework focuses on classifying anomalies in IIoT, it is important to know the monitored industrial process to recognize possible features or context information that help detect anomalies in IIoT systems. As Figure 4.2 shows, the implementation of the framework begins with a detailed description of how the CPS operates.

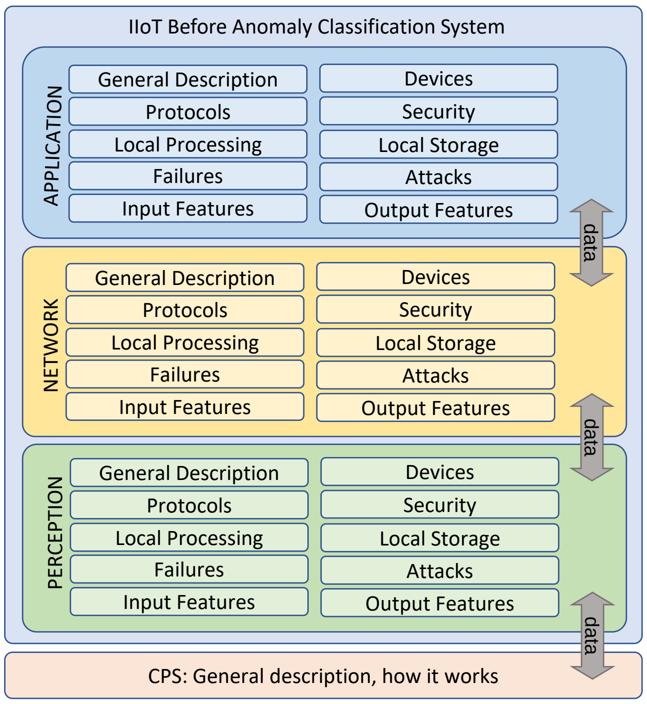

Figure 4.2: IIoT and CPS description. The first step to implementing the anomaly classification framework is to know in detail how CPS and IIoT work. Especially information flow. The figure lists the technical aspects that should be taken into account in each of the layers of the IIoT system.

4.1.2 Industrial Internet of Things

Industrial Internet of Things refers to a network comprised of devices and sensors that are interconnected and are used in various industrial applications [6–9]. This network is designed to collect and analyze real-time data, which is used to enhance efficiency, reliability, and decision-making processes. By utilizing advanced technologies like cloud computing, machine learning, and big data analytics, IIoT can optimize production processes, lower operational costs, and improve product quality [9, 12, 27]. Figure 4.2 is just a sketch to give developers an idea of what IIoT and CPS aspects they should consider; it is by no means a finished schematic. In this stage, it is expected that the developers of the anomaly classifier have a good understanding of the IIoT architecture implemented in the plant. This includes its layers, features, protocols, security, raw data, processed information, and how the IIoT system interacts with the industrial process. One way to achieve this is by creating a diagram illustrating the IIoT architecture, depicting information flows and the relationship between different parts. That is, transform Figure 4.2

into the diagram of the specific process of the plant that implements the framework.

4.2 Proposed Framework for Anomaly Classification in IIoT

This section delves into the intricate realm of Industrial Internet of Things (IIoT) anomaly classification, presenting a comprehensive framework designed to detect and categorize anomalies within industrial systems. As the IIoT ecosystem expands, robust anomaly detection mechanisms become increasingly critical to ensure operational efficiency, safety, and reliability. Our proposed framework leverages advanced machine learning algorithms and data analytics techniques to identify deviations from normal patterns in real-time, enabling preemptive actions and minimizing potential disruptions. By integrating various data sources and employing sophisticated classification models, this framework enhances anomaly detection accuracy and provides actionable insights for timely intervention. Through a detailed exploration of the underlying methodologies and practical applications, this section aims to equip practitioners and researchers with the knowledge and tools to implement effective anomaly classification systems in IIoT environments. The first step is to brainstorm and list the possible failures and attacks the IIoT system could suffer. These anomalies can be recorded in a table similar to Table 4.1, where the failures, attacks, and variables (available or not) that could help detect such anomalies are specified for each layer.

Table 4.1: Failures and Attacks in IIoT. Brainstorm and record here the possible failures and attacks the IIoT could suffer and the variables that could reflect those anomalies.

Layer	Failures	Attacks	Available Variables	New Variables
Anomalies by layers: cloud, edge, or sensing	Failures that can occur in each layer	Attacks that can occur in each layer	Available variables before Anomaly classification framework	New variable needed to detect failure or attack

Finally, in this section, additional features from IIoT must be identified for anomaly detection tasks. For instance, internal CPU data from edge devices can facilitate the detection of failures or attacks. With a good knowledge of the objects in each IIoT layer, it is possible to establish the information that can be used for anomaly detection and classification. At this point, there will be enough features to classify anomalies. However, those that put the system's security or privacy at risk or are complex or expensive to collect should be excluded. In conclusion, IIoT collects data to monitor the CPS and other variables to detect anomalies in its operation. This additional data may be obtained from the IIoT or the Cyber-Physical System. It is suggested to record the information collected in this section in Table 4.2, assessing the availability, confidentiality, cost, and complexity of collecting each variable.

Table 4.2: Features for Anomaly Classification. Select appropriate variables and evaluate the cost-benefit associated with using each of them.

Variable	Available (Yes/No)	Confidential (Yes/No)	Collection (Easy/Complex)	Collection Cost (Cheap/Expensive)	Anomaly	Decision (Include/Exclude)
Variable identified in Table 4.1	Available variable or a new variable	Is it a CPS confidential variable or not	How complex is its collection	How expensive is collecting the new variable	Which anomalies can be detected with this variable	Include or exclude the variable according to previous columns

4.2.1 Dataset

An anomaly is a value that differs significantly from the rest of the data. These differences may be caused by various factors, including malicious attacks, sensor malfunctions, or significant environmental changes [31–33]. This section describes some strategies for collecting the anomaly dataset to train the detection and classification models.

Anomalies

Every industrial plant and every IIoT system suffers from specific types of anomalies. Before collecting the dataset to train the anomaly classifier, the failures and attacks that will be detected must be identified, either because the company has suffered them or because it is likely to suffer them in the future. When failures are selected, the causes and their effects on the sensed variables should be considered since the anomaly detector will react to these effects. Some examples of failures are increases in CPU temperature, slow response, device disconnection, or an unexpected range of values sent by a miscalibrated or faulty device.

The security of an IIoT system depends on the implemented communication network security measures. Identification, authorization, and authentication mechanisms help the system ensure information privacy, integrity, and availability. Among the most frequent attacks on IIoT networks are denial of service, man-in-the-middle, Eavesdropping, Malware or Ransomware, social engineering, and zero-day attacks. Before collecting data, it is necessary to determine which attacks and failures the IIoT system needs to be protected against. This involves deciding which ones to include in the list and which ones to exclude. Some may be unlikely, while others may be too complex or expensive to address. Sometimes, alternative options, such as acquiring a policy, may be more practical. Table 4.3 is offered to facilitate the selection of faults and attacks. The pros and cons of including or not each anomaly are weighed.

Table 4.3: Anomalies in IIoT. Weight the cost and benefit of including or excluding each potential IIoT anomaly.

Anomaly	Probability (High, Medium, Low)	Risk (High, Medium, Low)	Variables	Training Dataset Complexity (Easy/Complex)	Decision (Include/Exclude)
Each anomaly identified in Table 4.1	How likely is it for this anomaly to occur?	If this anomaly occurs, how harmful is it?	Variables that could help detect the anomaly	Is it easy to collect data with this anomaly to train the model?	Include or exclude the anomaly according to previous columns

Training Data Collection

Autoencoder-type anomaly detectors can be trained on normal data. However, when algorithms are designed to classify outputs into more than two classes, the training data must encompass all relevant categories. Acquiring sufficient data samples to train such classifiers effectively poses a notable challenge. To overcome this issue, a combination of several strategies for data collection may be necessary. These strategies could include leveraging existing datasets, generating synthetic data to supplement existing samples, conducting targeted data collection efforts, and employing data augmentation techniques. Integrating these approaches makes it possible to assemble a diverse and comprehensive training dataset, facilitating the accurate training of classifiers for multiclass anomaly detection tasks. Some options for constructing a dataset with normal and abnormal data are described below.

- Use datasets available in public repositories. It is a quick and inexpensive option. However, the included features (variables) and anomalies are unlikely to efficiently describe the IIoT system to be protected with the classifier.

- Use software to emulate failures and attacks that could occur in the real IIoT. This option offers the possibility of generating many samples; however, building a simulation with all the details of the real IIoT may be challenging.

- Use historical data of the process with anomalies. It is possible that there are not enough samples with anomalies or that not all the types of anomalies that we want to classify are found.

- Artificially generate new samples from the available abnormal data. Currently, techniques such as generative neural networks (Generative Adversarial Networks, Variational Autoencoders, Restricted Boltzmann Machines) can be trained to generate artificial data from available data. However, it must be considered that training these models also requires a large amount of data.

- Simulate anomalies directly in the IIoT system while monitoring the industrial process. It is the most direct way to collect the data set, but it can be dangerous to simulate failures and attacks in the IIoT while the process is in production.

- Implement a testbed with devices similar to the real system and simulate failures and attacks there. It can be a less complex solution than programming a simulator, and if extra devices are available in the plant, it can also be not expensive.

Combining several of the above alternatives can lead to an optimal solution. Perhaps some flaws or attacks are challenging to implement in the physical world, and a simulation is the best option. The best data source is the industrial process and its IIoT system. However, these data contain few and varied anomalies, so other sources must be used to collect the data to train the detection and classification models. Another tool is a testbed that emulates the process, but the devices have a cost that can sometimes be very high. Artificial data could also be generated, but this requires much actual data to train the generator. Simulating a process to emulate anomalies is an unlimited source of data, but the complexity of programming a realistic simulation must be overcome. Finally, using public datasets is a cheap option, but finding a public dataset that matches the nature of the process of interest is unrealistic. In any case, the team implementing the framework must weigh the options, prices, and complexity to choose the most appropriate options. They must also use statistical techniques to validate that the generated data's distribution corresponds to the process's actual data. Use Table 4.4 to record information related to your data sources: repositories, simulations, data augmentation, testbed, or the actual system. Then, evaluate the best option for your anomaly classification system.

Table 4.4: Dataset for Training. Select anomaly data sources to train machine learning models.

Repository	Simulation	Data Augmentation	Testbed	Real System	Decision
Datasets available on public or private repositories or historical data	Software to simulate a similar IIoT and apply anomalies	Generative neural networks or apply anomalies to collected normal data	Implement a testbed with devices like on the real system to simulate anomalies and collect data	Apply anomalies directly on the real IIoT system.	After considering pros and cons of the options in previous columns, one or more of them are chosen.

Context Information

Context information is an aspect of any application, particularly in the industrial sector. In a few words, context awareness means using data from various sources to provide services that meet a user's specific requirements and expectations [61]. Here, it is crucial to identify any variables that might change based on the industrial process and can be useful in categorizing anomalies in IIoT. For instance, sounds can vary based on the frequency of the motor being monitored. In such situations, it may be necessary to include the industrial process variables as contextual information to detect anomalies. This contextual information can be added directly to the dataset or transformed through mathematical operations alone or in conjunction with IIoT variables. Table 4.5 can record possible context variables from CPS, environmental, commercial, production, or artificially created data.

Table 4.5: Context Information. Additional data from the CPS can improve anomaly classification.

CPS	External Data	Artificial Features	Decision
Variables from CPS to help anomaly detection	Environmental and production data, maintenance scheduling	New features created from equations and available data	Choose some variables that could be used as context information

4.2.2 Data Processing

The success of IIoT relies on accurate data from smart devices globally. Poor data quality can lead to incorrect decisions and declining user engagement. So, maintaining high data quality standards is essential for the widespread adoption and acceptance of IIoT services [4, 5]. Data processing transforms unstructured data into a structured format suitable for analysis. It ensures data quality, consistency, and error-free analysis, making it ideal for machine learning-based classification models [2]. Data processing involves feature elimination, handling missing values, removing duplicates, encoding non-numerical features, and normalizing the data [2]. Figure 4.3 illustrates the sequence of tasks that could be executed within data processing before using them to train the anomaly detection and classification models.

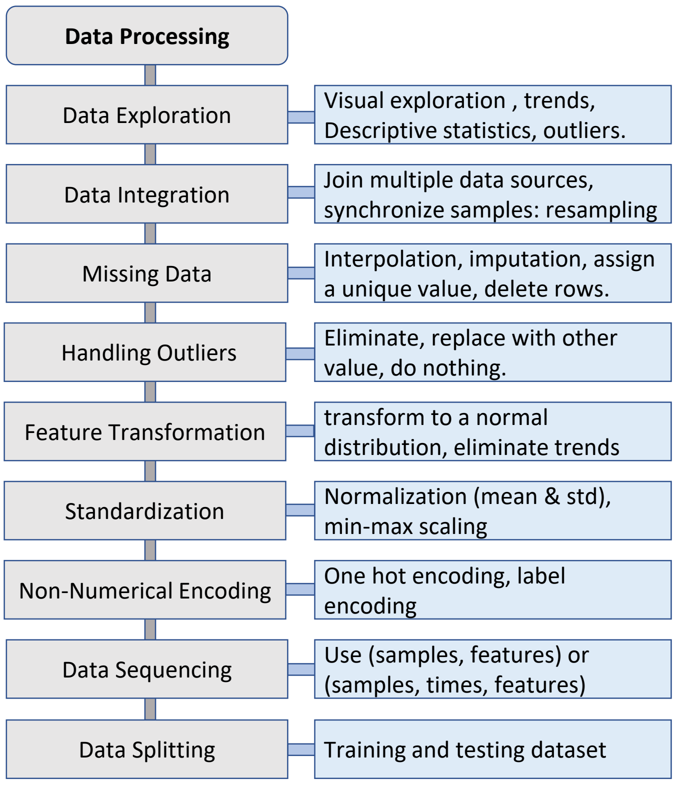
Data Processing
Data Exploration
Visual exploration , trends, Descriptive statistics, outliers.
Data Integration
Join multiple data sources, synchronize samples: resampling
Missing Data
Interpolation, imputation, assign a unique value, delete rows.
Handling Outliers
Eliminate, replace with other value, do nothing.
Feature Transformation
transform to a normal distribution, eliminate trends
Standardization
Normalization (mean & std), min-max scaling
Non-Numerical Encoding
One hot encoding, label encoding
Data Sequencing
Use (samples, features) or (samples, times, features)
Data Splitting
Training and testing dataset

Figure 4.3: Data Processing. Tasks to prepare data for training classifier and detector models.

Exploratory Data Analysis

It is a key step to perform an exploratory analysis at the beginning. This critical step involves using various statistical graphs and visualization techniques to carefully examine the dataset and identify any patterns, anomalies, or features that may be present. By doing so, the researchers can test their hypotheses and verify any assumptions they may have made about the data [46]. It is advisable to document all these findings to be used in the later stages of implementing the anomaly classification framework.

Data Integration

The Industrial Internet of Things (IIoT) relies on data from various smart devices. However, this data can be inconsistent and have structural discrepancies that must be addressed to provide seamless services. The primary objective of data integration solutions is to connect various data streams and fix any presentation inconsistencies that may occur [4, 64]. The first challenge in this phase is to synchronize variables collected independently and with different sampling rates. The question is how to merge these sequences into the same dataset: you should undersample the signals with a higher sample rate or oversample the signals with a lower sample rate. Alternatively, you should choose a middle point between these options. Programming languages such as Python and libraries like Pandas can automate the integration of all selected variables into a single dataset.

Missing Data

Missing values in datasets can create information gaps, which can harm our ability to understand a particular entity or phenomenon. This can result in incomplete knowledge or incorrect decision-making. Since these datasets are vital inputs in knowledge derivation processes, missing values can significantly reduce the overall data quality [4, 64].

If there are any missing values in a time series, they can be replaced with values within the device's detection range. There are a few methods to do this. One way is to interpolate between the neighboring data points, and another is to replicate data from the same set at the positions where data is missing. However, it is necessary to note that either method can introduce noise into the data set, making it more challenging for algorithms to learn from it [57]. It is important to differentiate between missing data due to system anomalies and missing data due to delays in data transmission, which can be considered normal for IIoT communication protocols.

Duplicate Removal

While building a model, it is essential to carefully select and process the data to ensure accuracy and effectiveness. One important step in this process is identifying and discarding duplicate instances in the dataset. These duplicates do not add significant value to the model and can often lead to biases and inaccuracies in the final results. The model can focus on the most relevant and informative data points by removing duplicates, leading to better performance and more accurate predictions. Therefore, it is crucial to identify and discard duplicate instances during the data preparation phase of building a model [2].

Handling Outliers

Data set management involves a critical component called data cleansing, encompassing many activities that ensure data quality and reliability. One of these tasks is identifying and managing outliers, which are values that fall outside the expected range within the context in which they were collected. The specific domain of knowledge is crucial in deciding whether an outlier should be eliminated, replaced with another value, or remain unchanged within the data set. A systematic procedure for removing outliers should be followed to ensure the ability of models to detect actual anomalies. In the technical literature, several options, such as clustering, machine learning, or statistics techniques, are offered to detect and replace values that should not be outside the expected range. Notably, cleaning outliers can only be performed on the training set. In contrast, the evaluation sets must remain unchanged regarding outliers to reflect the reality of the described process [4, 64].

Feature Transformation

Sometimes, some of the original data must be modified to meet specific analytical or modeling needs. The process aims to eliminate trends that have nothing to do with the study case, improve normality and linear relationships, handle outliers and interpretability, meet model assumptions, deal with zero-inflated data, facilitate convergence, and ensure positive values. It is important to note that the choice of transformation should be guided by the data's specific characteristics and the analysis's objectives.

Normalization and Min-Max Scaling

During the data processing, the dataset is normalized through data standardization to eliminate the influence of some features due to different dimensions [50]. When deciding whether to use normalization or min-max for scaling data, various factors should be taken into consideration, such as the characteristics of the dataset, data type, and model requirements [11, 12, 37]. On the one hand, normalization is used when assuming a Gaussian distribution, dealing with features at different scales, and algorithms sensitive to variance. On the other hand, min-max scaling is used with data bound to a range, algorithms requiring non-negative features, and ease of interpretation [12, 37].

Normalization adjusts values to have a mean of 0 and a standard deviation of 1, facilitating certain statistical analyses and training machine learning models. Min-max scaling adjusts the values of the data to a specific range $[a, b]$, commonly to $[0, 1]$ or $[-1, 1]$, which can help compare measures that have different units or scales [12, 37].

Non-Numerical Features Encoding

To make it possible for the reduction and machine learning algorithms to process the data, it becomes necessary to convert non-numerical features into numerical ones. Machine learning can handle categorical variables using label encoding and one-hot encoding methods. The choice between these methods depends on the specific dataset and the machine learning algorithm. While label encoding is simpler and more space-efficient, it may introduce an arbitrary order to categorical values. One-hot encoding avoids this issue by creating binary columns for each category, but it can lead to high-dimensional data [2].

Data Sequencing

The sliding window technique in time series analysis transforms a dataset into a format suitable for machine learning models, especially deep learning [11, 37, 48]. Transforming time series data from [n-samples, n-features] to [n-samples, timesteps, n-features] is crucial for several reasons explained in Chapter 2, such as improved learning dynamics, efficient data utilization, flexible forecasting, feature utilization, model input requirements, and temporal dependency capture [9, 48]. Choosing the right window size for anomaly detection is crucial. It should be large enough to capture patterns but not too large to dilute anomalies. No universal solution exists, so experimentation is necessary to balance relevance and computational efficiency [11, 37].

Data Splitting

When working with IIoT networks, splitting the initial data into two sets is important. This process is called data splitting. The first set is the training set used to teach the model. The second set is called the test set, and it is used to evaluate the final performance of the trained model. To get an accurate assessment of the trained model's effectiveness, it is crucial for the test data set to maintain a similar distribution of classes as the training set. By doing this, you can replicate the actual scenario of IIoT networks [2].

4.2.3 Feature Selection

Unnecessary features present in data lead to computational complexity and significantly decrease the precision and effectiveness of classification techniques [2, 56]. Several techniques and criteria can be used to select the variables or features that should remain in the dataset. Among the most important are:

- Understanding the problem domain helps identify the variables in the dataset and the variables that should be discarded because they vary with other variables that are not included or related to the study case.
- Exploratory data analysis for understanding the nature of variables, recognizing relevant patterns, variable distribution analysis, identification of outliers, discovery of hidden patterns, and nonlinear relationships.
- Dimensionality reduction techniques can be split into feature extraction and feature selection. Feature extraction reduces data dimensionality by projecting original features into a new low-dimensional feature space [2, 56]. Feature selection aims to choose a smaller set of features that can predict class labels more effectively and has a lower dimensionality than the original feature space. Moreover, one must consider whether the data are time series and the method is designed to work with them.
- Raw data sets often contain multiple correlated features. Including all these features in a learning model can significantly increase the computational burden, leading to long training times [12]. To avoid this, feature selection is performed by analyzing the correlation between the features through mutual information, Pearson, spearman, and others. Then, you should remove highly correlated features and discard those with zero variance.
- Automatic variable selection through recursive feature elimination, using decision tree models, or regularization-based methods such as LASSO or Ridge. It is also helpful to evaluate the model's performance with different variables (Cross-validation) or create new features from those available.

Generally, the best option is to combine several techniques mentioned above to decrease the number of features that describe an object, making it easier to understand anomalies. Consider creating a table like Table 4.6 to track variables that could be removed from your dataset. It is recommended that anomaly detector and classifier models be trained both with and without these variables to ensure that removing them improves the performance of the models.

Table 4.6: Feature Selection. Apply a combination of techniques to select features in the dataset.

Exploratory Analysis	Specific Domain Knowledge	Statistic Analysis	Dimensionality Reduction Techs	Recursive Feature Elimination	Decision
Recognizing relevant patterns	Identify variables not related to the study case	Mutual information, Pearson, Spearman, Lasso, Ridge, elastic nets.	Feature extraction and feature selection	Evaluate the model's performance with different variables	Choose the set of variables after trying the strategies suggested in each column.

4.2.4 Data Balancing

When datasets are unbalanced, machine learning models may exhibit a bias towards the majority class, which can result in no optimal performance for the minority class. This can be especially challenging for applications such as anomaly detection and predicting rare events, where the minority class holds greater significance [38, 68, 69]. In multiclass classification, methods to address class unbalance include resampling, classifier adaptation, ensemble strategies, and cost-sensitive techniques [38, 70]. An explanation of each technique is found in the chapter 2.

A way to deal with a scarcity of available data is to generate synthetic data using a generative model based on the initial dataset. This process is called data augmentation, which involves using human-labeled data to produce new data points similar to the original ones [30, 42]. Resampling methods directly answer the unbalanced challenge among the spectrum of augmentation strategies. The Synthetic Minority Over-sampling Technique (SMOTE) generates synthetic samples to broaden the decision region of the minority class [30, 42, 57, 69].

While resampling adjusts the balance between classes, simulation-based augmentation further enriches the model's training environment, simulating conditions unseen in the real world to prepare models for a broader set of challenges. On the one hand, simulating anomalous conditions through reliable and high-fidelity simulation data can offer low-cost training data and address the problem of insufficient samples [30, 42]. On the other hand, GANs are Generative Adversarial Networks used for data augmentation in domains where data is limited or expensive. They consist of a generator and a discriminator that are trained together. The generator produces data similar to real data, while the discriminator distinguishes between real and generated data [30, 42].

The technique or combination of techniques you choose to balance the number of samples in each class will depend on the nature of the data and the resources available to expand the samples in the minority classes. If you have a two-stage classifier, where the first is trained only with normal data and the second with balanced data, subsampling the majority class may be the most advantageous solution without losing the model's generalization. Table 4.7 summarizes the aspects to be weighed to decide the techniques used to balance the dataset.

Table 4.7: Data Balancing. Choose between undersampling or oversampling to balance classes or adjust models for unbalanced datasets.

Resampling	Data Augmentation	Adapted Models	Decision
Undersampling majority class or oversampling minority class or boths	Create new samples for minority class: by collecting or artificially	Adapt machine learning models to work with unbalanced datasets.	Choose one or more options from previous columns

4.2.5 Anomaly Detection

Anomaly detection (AD) in IIoT is a process that identifies patterns in time-series data that deviate from the system's normal behavior [9]. AD should focus on detecting, isolating, and recovering from attacks, identifying service vulnerabilities, addressing transmission problems between nodes, and controlling packet growth to prevent service breaches [22]. Chapter 2 describes some techniques for anomaly detection.

An approach suggested in this framework is to employ an unsupervised method for spotting anomalies, in which the model is trained only with normal data. This technique facilitates the identification of previously unknown anomalies. Autoencoders are an example of this approach. They consist of an encoder and a decoder, which compress input data and regenerate it from the latent space representation, respectively [9, 53]. When using an unsupervised model to detect anomalies, the absence of labels is made up for by setting a threshold value that distinguishes normal data from anomalies. This is done by assigning a score, which is then used to determine if the data is normal or an anomaly. Various techniques are employed to calculate the optimal threshold value, balancing sensitivity and specificity [53, 54]. Common approaches such as fixed threshold, mean plus standard deviation, iterative method, and Dynamic Thresholding were explained in Chapter 2.

4.2.6 Anomaly Classification

This framework proposes a two-stage approach to anomaly classification. In the first stage, a detector differentiates between normal data and anomalies. The second stage is triggered if the samples exceed the threshold value to classify the anomaly. The classifier is designed to identify normal data and various failures and attacks. It provides one of three labels as output: normal, fault, or attack. However, when it is used to predict, it only receives data sequences classified as anomalies by the detector. Therefore, if the classifier identifies one of these sequences as *normal*, the application generates an *event* alert. This alert will help the human operator manually verify the alert's origin. Although the framework does not integrate it, sequences labeled *Failure* or *Attack* could pass through a new classifier that specifically indicates what type of failure or attack they are.

Metrics

In binary and multiclass classification, we use evaluation metrics to measure various characteristics of an algorithm's performance. These metrics allow us to assess the algorithm's accuracy, precision, recall, F1 score, and the Receiver Operating Characteristic (ROC) Curve and Area Under the ROC Curve (AUC-ROC) [34, 38, 73]. You can find a detailed explanation and equations for each evaluation metric in Chapter 2. These metrics are useful in obtaining valuable information about the capabilities and limitations of the classification algorithm, which can help determine its suitability for a particular task.

The metric you select depends on the classification task's objectives and the dataset's characteristics. Metrics should be chosen taking into account classification type, efficiency in computation, understandability and discriminative power, depth of performance insight, and attention to underrepresented classes [34, 38, 39, 74]. The evaluation metrics are designed to assess the generalization capability of the trained classifier, mainly when tested with previously unseen data. To optimize the anomaly detector and classifier, the selected metrics must be calculated for each model separately and for both working together. Accuracy is not recommended if unbalanced data sets are used since this metric favors the majority class. Instead, precision, recall, F1-score, or AUC-ROC are suggested.

Hyperparameter Optimization

Optimization is an important task in anomaly classification. It involves tuning model parameters, adjusting learning rates, and selecting appropriate algorithms. This process requires precision, domain knowledge, and practical considerations. Doing so can increase the probability of correctly identifying anomalies, thereby reducing false positives and negatives. Selecting the appropriate hyperparameter optimization technique depends on several factors, such as the particular problem, the computational resources available, and the kinds of hyperparameters involved. Although grid search and random search are simple to execute, approaches like Bayesian optimization, evolutionary algorithms, and Hyperband offer more advanced and perhaps more effective ways to explore the hyperparameter space [77–79].

4.2.7 Alarms

The anomaly classifier alerts the system operator in case of normal operation, failures, or attacks. The purpose of classifying anomalies is to inform the relevant staff. The maintenance department handles device failures, while the IT staff handles malicious attacks. If the classifier labels an anomaly as normal, the system generates an *event* alert, and the system operators will manually reclassify it as normal, attack, or failure. The sequences that arrive at the classifier should be included in the dataset to retrain the model and improve its performance.

This chapter comprehensively outlines the framework for anomaly classification in the Industrial Internet of Things (IIoT), focusing on integrating Cyber-Physical Systems (CPS) with IIoT technologies to create a smart factory environment. Key aspects discussed include the importance of understanding the CPS and IIoT architecture, which forms the foundation for effective anomaly detection. The chapter identifies potential failures and attacks, emphasizing the need for feature selection to enhance detection accuracy. Data collection and processing are highlighted as crucial steps, with techniques for handling missing data, duplicates, outliers, and data scaling thoroughly explored. The importance of balancing datasets is addressed alongside advanced machine-learning models for anomaly detection and classification. Evaluation metrics and optimization techniques are discussed to ensure robust model performance. The chapter concludes with practical insights into generating alarms and integrating the framework into real-world industrial settings, providing a holistic approach to enhancing operational efficiency, safety, and reliability in IIoT environments.

Chapter 5

Framework Implementation Results

This chapter delves into the practical implementation and evaluation of the IIoT Anomaly Classifier Framework within a testbed environment. Our experimentation involved an emulated Cyber-Physical System (CPS) represented by a four-belt conveyor system and monitored by an Industrial Internet of Things (IIoT) system through a cloud platform. This setup was chosen to simulate real-world industrial conditions closely and rigorously test the framework's capability to detect and classify anomalies. The novel contributions of this chapter are multi-faceted, emphasizing enhancements in anomaly classification within Industrial Internet of Things systems:

- Innovative implementation for anomaly classifier framework. We implemented a new framework designed to identify and classify anomalies in IIoT systems, leveraging real-time data and advanced analytics.

- Integration of detection and classification models for improving anomaly resolution. This dual-model approach is designed to refine the accuracy and efficiency of identifying and classifying anomalies in IIoT systems.

- Enhancing anomaly classification through rigorous data processing. By carefully preprocessing data, the framework improves the classification model's overall performance, making it better at identifying and categorizing anomalies in IIoT systems.

- Practical implementation of context information to improve the anomaly classification system. By integrating contextual data, such as the operational states of the cyber-physical systems, the classifier can make more informed decisions about the nature of the anomalies detected. This contextual awareness allows the system to differentiate between anomalies from failures or cyber-attacks.

- Emulated testbed for realistic validation. This chapter details constructing and utilizing an emulated testbed that simulates an operational industrial environment. This setup is critical in validating the efficacy of the Anomaly Classifier Framework under realistic conditions.
- Comprehensive evaluation using real-world scenarios. Our experiments reflect real-world scenarios, ensuring the findings apply to industrial operations. The framework's performance is evaluated across various simulated conditions to assess its robustness and reliability.
- Integration of IIoT with deep learning models. We examine the successful combination of IIoT technologies with deep learning methods to improve monitoring and anomaly detection, representing a significant advance in machine learning for anomaly classification in IIoT.

The results of these contributions are discussed in detail below, providing valuable insights into the framework's practical applications and its potential to improve operational safety and efficiency in industrial environments. This chapter is structured to present the implementation results of the proposed IIoT anomaly classification framework clearly and comprehensively. We begin with an introduction to the experimental setup, detailing the configuration of the testbed, which includes an emulated Cyber-Physical System (CPS) and an Industrial Internet of Things (IIoT) system. Following this, we delve into the specific results of the framework's performance, analyzing key metrics such as detection accuracy, false positive rates, and computational efficiency. Each subsection within this analysis highlights different components and stages of the implementation, including data preprocessing, model training, and validation processes. Comparative analyses are provided to benchmark the proposed framework against existing anomaly detection methods, demonstrating its relative advantages and areas for improvement. The chapter also includes practical case studies and real-world scenarios to illustrate the framework's effectiveness in operational settings. We conclude with a synthesis of the findings, discussing their implications, potential improvements, and suggestions for future research to enhance further the robustness and applicability of the framework in diverse industrial contexts.

5.1 Smart Factory

To test the effectiveness of the Anomaly Classifier Framework and following the recommendation given in Table 4.4 of Chapter 4 "Framework for Anomaly Classification," a testbed was set up with an emulated Cyber-Physical System (CPS) and an Industrial Internet of Things (IIoT) system. The CPS used in the testbed was a four-belt conveyor system created to emulate real-life scenarios. The IIoT system monitored the CPS through a cloud platform, allowing real-time data analysis and reporting. Testing the Anomaly Classifier Framework in this way lets us assess its ability to detect anomalies in a real-world industrial setting.

5.1.1 Cyber-Physics System

A cyber-physical system composed of four conveyor belts has been emulated to implement the anomaly classification framework. Figure 5.1 shows their distribution in the industrial plant. Depending on the product transported, the system follows one of three recipes: in the first one, only motors in conveyor belts A and B are activated, and their respective drivers generate speeds proportional to 40Hz, while the second recipe energizes motors in conveyor belts A, B, and C at 50Hz and the third recipe powers the four ones at 60Hz. The change of recipe or the shutdown of all motors is controlled from a graphical interface (Human Machine Interface, HMI).

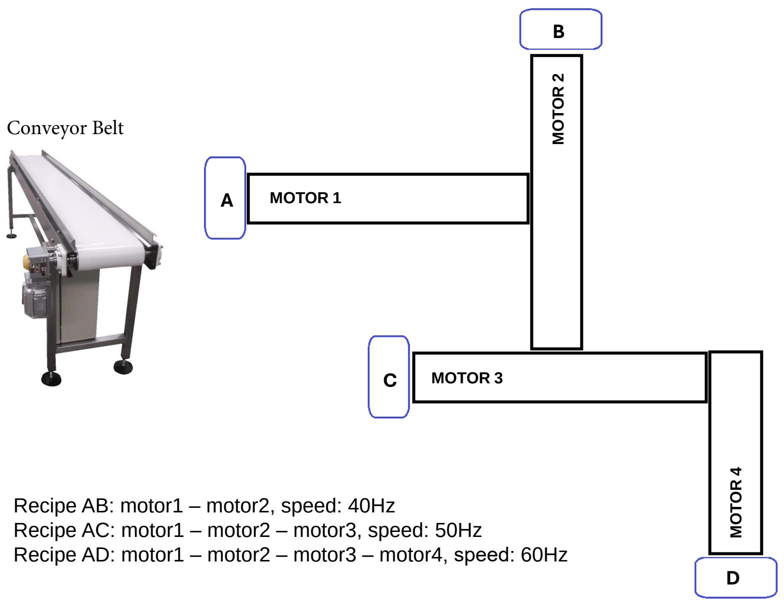
Conveyor Belt
B
MOTOR 2
A
MOTOR 1
C
MOTOR 3
MOTOR 4
D
Recipe AB: motor1 – motor2, speed: 40Hz
Recipe AC: motor1 – motor2 – motor3, speed: 50Hz
Recipe AD: motor1 – motor2 – motor3 – motor4, speed: 60Hz

Figure 5.1: Emulated CPS to evaluate Anomaly Classification Framework. Three recipes control four 3-phase AC motors from an HMI.

Four testbeds have been used to emulate the conveyor belt system; Figure 5.2 shows one of them. Each comprises an AC motor, a speed controller, an S7-300 programmable logic controller (PLC), and a Human Machine Interface (HMI). The four testbeds are connected through an Ethernet network implementing Profinet protocol.

Figure 5.3 shows a diagram of the electromechanical devices involved in the CPS. A PLC controls each motor through a speed controller. All of them are interconnected by a Profinet network and communicate with a supervision system and an HMI. From this screen, the different recipes are activated, and the main operating parameters of the system are displayed.

5.1.2 IIoT

An IIoT Industrial Internet of Things system has been implemented to monitor the CPS from a cloud application. In Figure 5.4, the CPS described in the previous section is represented only with four motors and an HMI. IIoT comprises four Raspberries PI 3B, acting as edge devices, several low-cost sensors on each motor to record sound and vibrations, and sensors near two edges to record the humidity and temperature in the plant. Each sensor communicates with an ESP32 microcontroller, which is responsible for collecting measurements and sending them to its respective edge device through

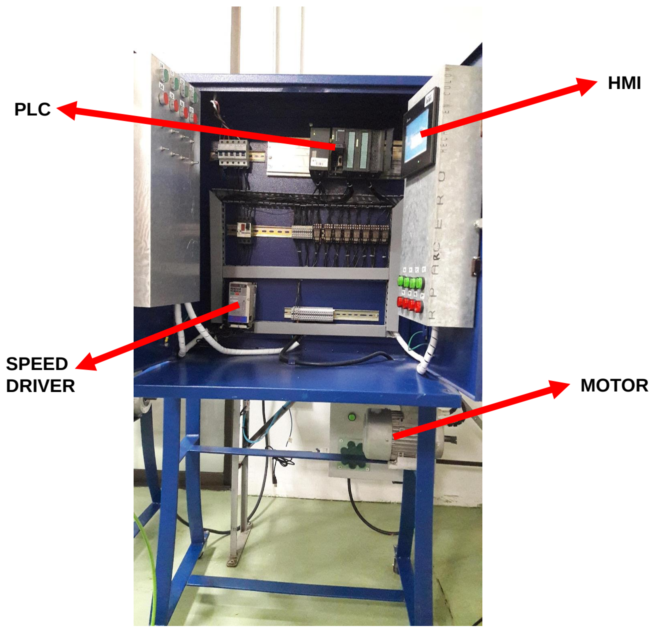

Figure 5.2: Testbed used for emulating the CPS. It is composed of a three-phase motor, speed driver, S7-300 PLC, and Delta HMI.

MQTT (Message Queuing Telemetry Transport) protocol. Additionally, an edge device communicates via MODBUS RTU with the HMI to obtain information about the motors. On each edge device, data is processed locally before being sent to a monitoring and control application deployed in the AWS cloud. This IIoT system will also collect data to detect and classify anomalies while monitoring the CPS.

5.2 Dataset

The CPS-IIoT suite was described in the previous section, where three failures and two cyber attacks were simulated. Data from the motors, internal data from each edge device, and current consumption were collected. The details of this procedure are described below.

5.2.1 Anomaly Generation

In this framework implementation, we tried to differentiate between failures and attacks. The following describes the failures and attacks executed on the testbed.

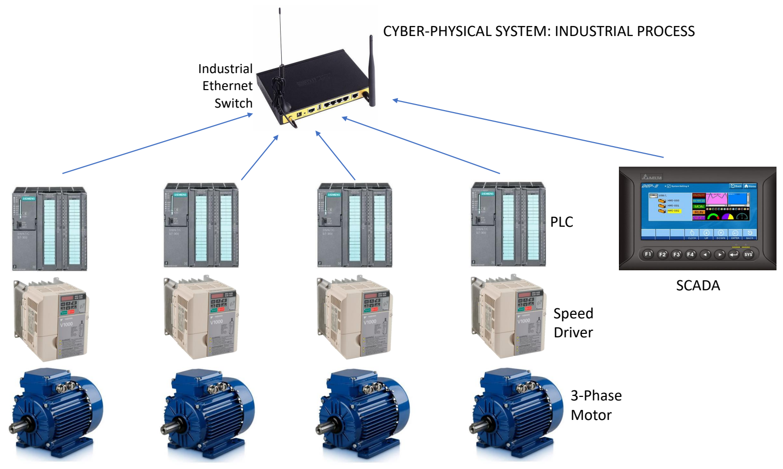
CYBER-PHYSICAL SYSTEM: INDUSTRIAL PROCESS
Industrial Ethernet Switch
PLC
SCADA
Speed Driver
3-Phase Motor

Figure 5.3: Electromechanical devices involved in the CPS. Three-phase motors are controlled and monitored through speed drivers connected to PLCs and an HMI.

- High-temperature failure (F1) was simulated by turning off the fan that cools the edge device. A 24-watt AC fan was placed 20 centimeters above the Raspberry Pi. Every time an anomaly was generated on the respective edge device, a digital signal was manually activated to tag that data with the specific type of anomaly.

- Miscalibrated Sensor (F2). To simulate the miscalibration of the motor sound sensor, the actual measurement value was increased by 20%. This was done in the collected data, as miscalibration was not directly simulated on the sensor. This failure can last for several minutes.

- Sensor disconnected (F3). A sound sensor disconnection fault was added, turning off the sensor and its respective ESP32. The dataset reflects this failure as zero values in the respective feature. This failure is not used during training but to evaluate the model with previously unseen anomalies.

- A data injection attack (A1) was executed by spoofing the ESP32 connected to the motor sound sensors on each edge device. For this, a Python script was executed from a laptop connected to the IIoT network. The script listened to the MQTT channel and stole identification data from each end device, then used this data to spoof the microcontroller and send the MQTT broker random data between 0 and the average value of the motor's sound signal. The communication channel used is Wi-Fi, between the sensors and the edge device and between the edge device and the cloud application.

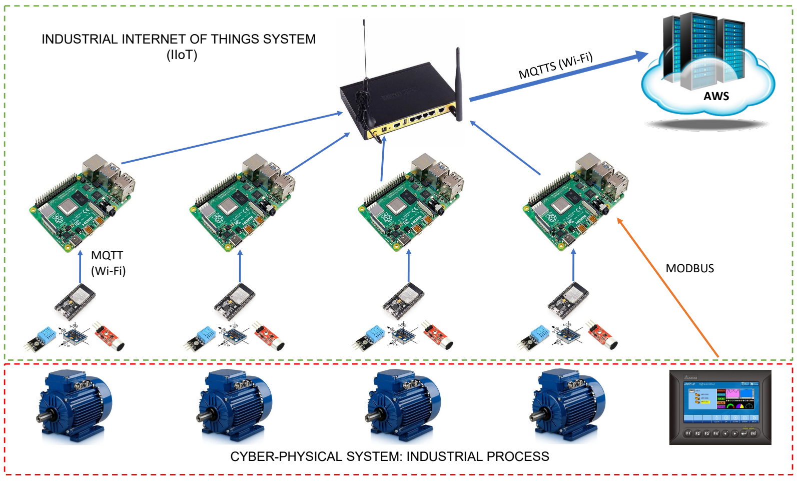
INDUSTRIAL INTERNET OF THINGS SYSTEM
(IIoT)
MQTTS (Wi-Fi)
AWS
MQTT
(Wi-Fi)
MODBUS
CYBER-PHYSICAL SYSTEM: INDUSTRIAL PROCESS

Figure 5.4: An IIoT three-layer (perception, network, application) model is used for anomaly classification. CPS is represented here with four three-phase motors and an HMI.

- Denial of Service (DoS) attack (A2) was launched on the MQTT brokers running on all edge devices. A Python script on a laptop connected to the local network subscribed 1024 clients to the topic "#." It published short messages to the Mosquitto MQTT broker every second, thus saturating the communication channel. As in the previous case, the channel used is Wi-Fi

As mentioned, a digital signal was activated each time an anomaly was generated on the Raspberry PI. Thus, the sensor measurements and the CPS data were stored with the respective timestamps and labels that identified the anomaly as failure (‘F1’, ‘F2’ or ‘F3’), attack (‘A1’ or ‘A2’), or normal (‘N’). Table 5.1 represents the assessment of the selected anomalies, based on Table 4.3 of Chapter 4.

5.2.2 Data Collection

Data were collected for 10 days between 6:00 am and 8:00 pm. At intervals that vary between two and four hours. Each edge device collected all the measurements and, one by one, added a column for the anomaly type (‘N’: normal; ‘F’: failure, ‘A’: attack), another column for the type of failure (‘N’:none, ‘F1’: temperature, ‘F2’: miscalibration, ‘F3’: disconnection) and other one for the type of attack (‘N’: none, ‘A2’: ‘A1’: DoS, Data Injection.

Table 5.1: Anomalies in IIoT according to framework description.

Anomaly	**Probability (High, Medium, Low)**	**Risk (High, Medium, Low)**	**Variables**	**Dataset (Easy/ Complex)**	**Decision (Include/ Exclude)**
High-Temperature Failure	Medium	Medium	CPU temperature, frequency.	Easy	Include
Miscalibrated Sensor	Medium	Medium	Sensor data	Easy	Include
Sensor Disconnected	Medium	Medium	Sensor data, WiFi packets.	Easy	Include
Data Injection Attack	High	High	Bytes sent or received over WiFi, CPU load.	Easy	Include
Denial of Service Attack	High	High	Bytes sent or received over WiFi, CPU load.	Easy	Include

Data from CPS

Through the HMI, an edge device collects data from all the motors in the conveyor system. This data is stored on this device in CSV files and includes anomaly-type labels. In the data processing phase, these files are shared with the other edge devices to build the datasets. Data collected from the CPS includes the motors' frequency, voltage, current, and power and the recipe executed on the conveyor belt system. Sound, accelerometer, and gyroscope measurements were collected from the motors through MQTT protocol. In this implementation, we use the frequency of each motor in the CPS as context information to improve anomaly detection in the IIoT system.

Internal Data from CPU Edge device

Internal data from each Raspberry PI was collected with the RPi-Monitor [190] app and saved into CSV files using a Python script. The CSV file includes measurements of internal temperature, frequency, CPU load for 1, 5, and 15 minutes, voltage, free and available memory, and packets sent and received over WiFi. Additionally, the current consumed by the CPU, collected through an INA219 sensor, was added to the CSV file.

Other information collected

Relative humidity and environment temperature measurements were collected at two different locations in the plant. The data was sent over MQTT to the nearest brokers, who stored it in CSV files. Table 5.2 shows details of raw data collected by the IIoT system to classify anomalies. It should be noted that the data collection rate is variable since it depends on network congestion in ESP32 and Raspberries PI.

Table 5.2: Data Collection Details. Raw data were collected from different sources with different sampling rates and stored in separate files.

Location	**File**	**Variables**	**Rate**	**Description**
CPS	modbus.csv	frequency, voltage, current, power, recipe	0.1s	Data from all the motors in the conveyor system
CPS	sound.csv	sound	0.001s	Sound of motor
CPS	mqtt_accel.csv	accelerometer, gyroscope	0.001s	Motor vibrations
Edge Device	internal.csv	Temperature, frequency, CPU load for 1, 5, and 15 minutes, voltage, free and available memory, packets via WiFi, current	1s .. 10s	Internal data from edge device collected with RPI monitor App.
Building	mqtt_temp.csv	humidity and temperature	10s	Environment data

5.3 Data Processing

Data preprocessing is a critical stage in machine learning, as it considerably impacts the effectiveness and efficiency of the learning process. Raw data often contains inconsistencies, errors, or outliers that can negatively affect the model's performance. Preprocessing involves cleaning data by handling missing values, removing duplicates, and correcting errors. The primary data processing tasks on the collected dataset are described below.

5.3.1 Data Exploration

Data exploration ensures that subsequent steps in data science and machine learning are informed, effective, and tailored to the data's specifics. It involves understanding the data's patterns, irregularities, underlying structure, and critical characteristics through visual and quantitative methods. Figure 5.5 shows some graphics obtained during visual data exploration. The figure shows, for example, that the features associated with packets sent and received over WiFi have increasing trends, positive or negative, or that the motor's sound is strongly related to its frequency. Visual exploration of the data also allows us to determine outliers and normal ranges of the data.

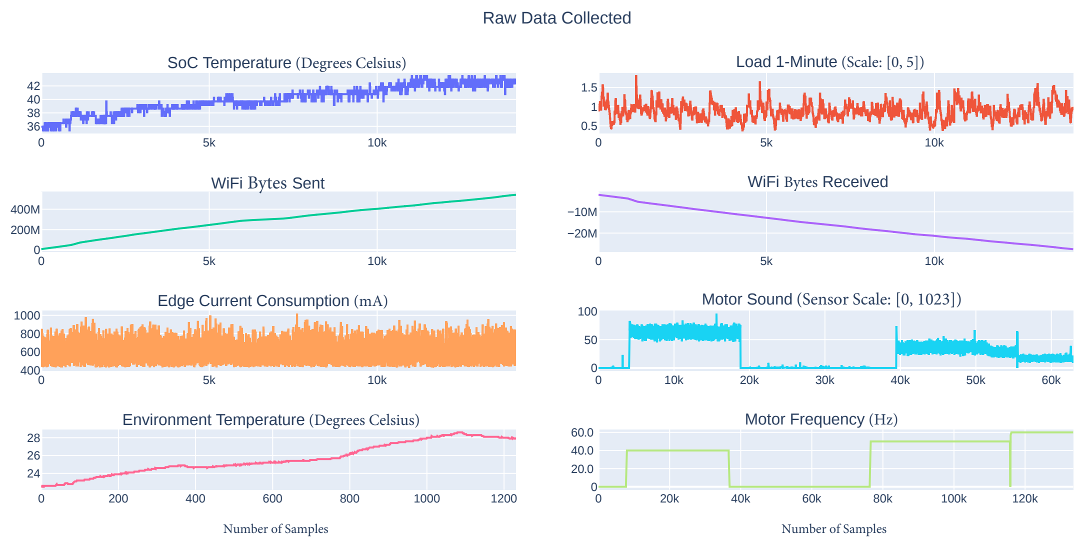

Figure 5.5: Raw Data Collected. The plots display the appearance of some of the signals that are gathered by the IIoT system.

Likewise, the statistical distribution of the collected data was determined using the Fitter library and a Python script. Table 5.3 showcases the relationship between various variables and the statistical distributions of each one. It indicates that none of the features show a normal distribution, which is frequently assumed in statistical modeling due to its symmetric bell-shaped curve defined by a mean and variance. The environmental temperature and humidity, along with CPU-related metrics such as load, current, temperature, and the number of packets sent, are described by a lognormal distribution. This distribution is chosen when the data are positively skewed, meaning there is a longer tail on the right side of the distribution. Powerlaw distribution is more suitable for the CPU frequency, voltage, and number of packets received. This distribution is characterized by the prevalence of more minor occurrences and the rarity of larger ones, with a slower decay rate in the tail, indicating a higher probability of extreme values than other distributions. Memory variables, such as available and free memory and CPU load over the last minute, correspond to a Cauchy distribution. This distribution is notable for its heavy tails, which imply a higher likelihood of observing extreme values. An exponential distribution is mentioned for motor current and sound,

Table 5.3: Statistical Distribution of Features. None of the features correspond to a Gaussian distribution, which determines some data processing actions.

Distribution	Features	Description
Lognormal	indoor_temp, load15, indoor_humidity, soc_temp, load5, wifi_sent, cpu_ current	Random variable whose logarithm is normally distributed. It is used in various fields to model non-negative data with a rightward skew.
Powerlaw	cpu_freq, cpu_volt, wifi_receiv,	Large occurrences are extremely rare, but small ones are quite common
Cauchy	load1, memory_free, memory_avail,	The tails of a Cauchy distribution decrease polynomially.
Exponential	motor_sound, motor_current	Describes a process in which events occur continuously and independently at a constant average rate.
Uniform	motor_freq, motor_volt, motor_recipe	All intervals of the same length have an equal probability of being observed

typically used to model the time between continuous and independent events occurring at a constant average rate. Lastly, motor frequency, voltage, and the recipe in CPS are uniformly distributed. The uniform distribution assumes that all values within certain bounds are equally likely to occur, making it a natural choice for modeling scenarios where no value within the range is any more likely than another. In the context of IoT, real-time data generation is usually rapid, and many data sources are available. This often results in issues related to the time alignment of these data sources. Table 5.2 displays various sampling rates that range from 1 millisecond to 10 seconds, depending on the data source. Moreover, it is necessary to consider that data did not arrive at the edge device consistently. If the communication channel was flooded or the device was occupied with other activities, samples were obtained and stored at a lower frequency. For instance, the internal CPU data of the Raspberry Pi could be acquired from the RPIMonitor application either every second or every 10 seconds. The abovementioned problem makes merging all the data into one dataset even more challenging.

The study case involves changing the sampling rate of time series data from different sources to the same rate. A fixed rate of 5 samples per second was selected, translating to a sampling rate of 0.2 seconds. To achieve this, subsampling has been carried out on the data gathered from the motor, including sound and vibration. Additionally, oversampling has been performed on the internal data of the CPU, as well as on room temperature and humidity, applying linear interpolation. This sampling rate was chosen to lose the minimum number of samples possible, considering that even signals sampled every 0.1s sometimes arrived at their destination every 0.3s or slower due to the conditions mentioned in the previous paragraph.

5.3.2 Data Integration

Another aspect to consider is that not all data sources were available simultaneously. Figure 5.6 represents with timelines the availability of each file integrated into a single dataset for one of the edge devices on the first three days of collection. In the first session of day 1, intermittencies are observed for most of the files. This is probably due to adjustments and tuning of the testbed. So, all data collected before 18:00 was discarded. Each data collection session was stored in a different CSV file. In the graph for day two,

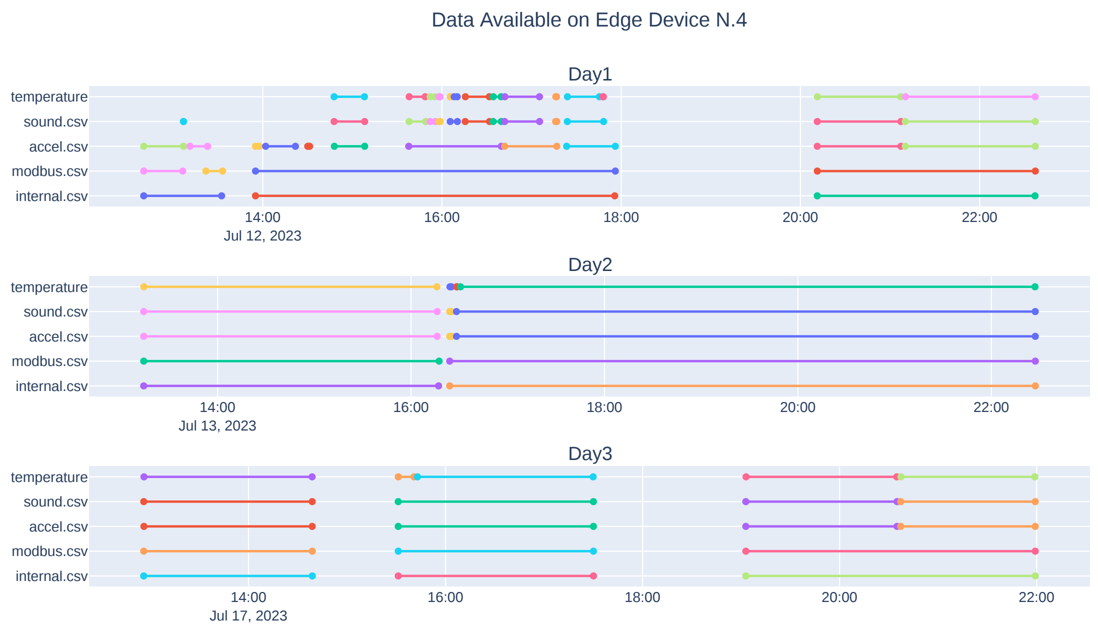

Figure 5.6: Raw Data Integration for Edge device N.4. Data was collected in separate CSV files, which were later integrated.

an interruption is observed after 16:00; although brief, thousands of samples stopped being collected for some variables, so day two is taken as two separate sessions. In other parts of the graph, it is observed that the end of a line overlaps with the beginning of the following line in a different color; this indicates that a few data are missing, which, depending on their quantity, have been reconstructed using linear interpolation or hot-deck imputation, as described below.

5.3.3 Missing Data

When working with deep learning, it is vital to handle missing data effectively since it can significantly impact the accuracy and reliability of model predictions. We used linear interpolation and hot-deck imputation to fill in missing data to address this issue. Linear interpolation involves generating a new value between two existing values by taking their average. We used this technique when only a few samples were missing, such as when oversampling was applied to ensure a consistent sampling rate of 0.2s across the whole time series. Besides, we used hot-deck imputation when there were delays in collecting data due to congestion in the communication channel or microprocessor issues, causing the absence of many samples. This technique involves copying a range of well-conformed data of the same signal into the interval where samples are missing. In Chapter 2, these techniques are explained.

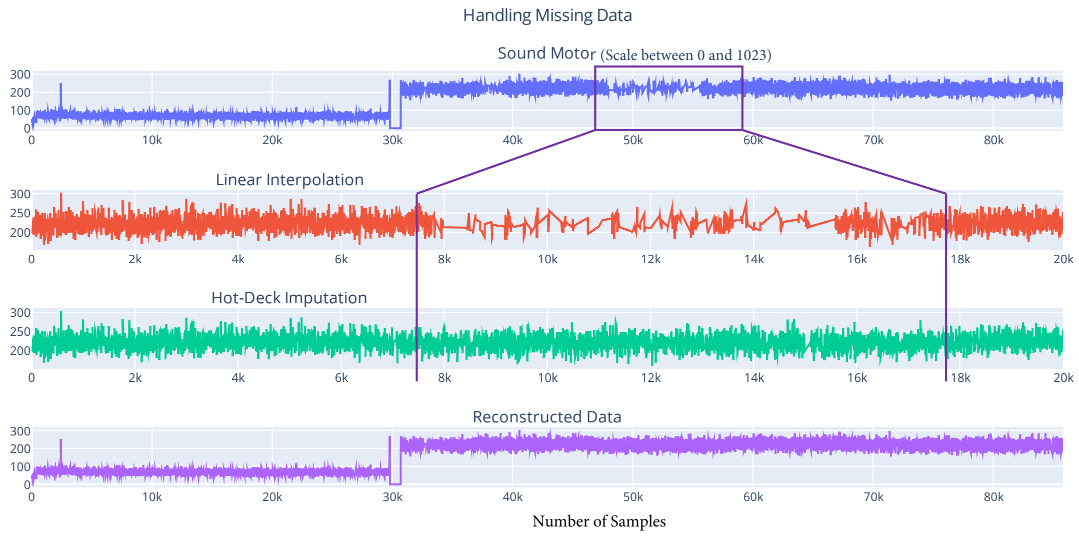

Figure 5.7: Handling Missing Data. Two methods to replace missing values are linear interpolation, which creates a new value between two samples, and hot deck imputation, which copies a range of data from the same series.

Figure 5.7 shows a section of the motor sound where fewer samples than the average rate were collected (above). This section is zoomed in, and the graphs in the center show what the signal would look like if the data were aggregated using the linear interpolation technique (red) and hot-deck imputation (green). It could be concluded a priori that for long stretches of data, hot-deck imputation better preserves the signal's shape (below).

5.3.4 Feature engineering

Feature engineering is a process that involves using domain knowledge to create new features or modify existing ones to increase the predictive power of a machine learning model. Practical feature engineering can significantly improve the performance of the model. In this case, to eliminate the ever-increasing trend of bytes sent and received over WiFi, the delta value of the signals has been decided to be used. This means taking the difference between each value and the value that comes immediately after (in the case of bytes sent) or before (in the case of bytes received). Figure 5.8 provides a representation of the original signals (a. and b.) and the delta values of each signal (c and d.). Another reason for performing this transformation was the wide range of data. The difference between the initial and final data values is of the order of Megabytes, which hides in the scaled data the subtle patterns that indicate the occurrence of an anomaly.

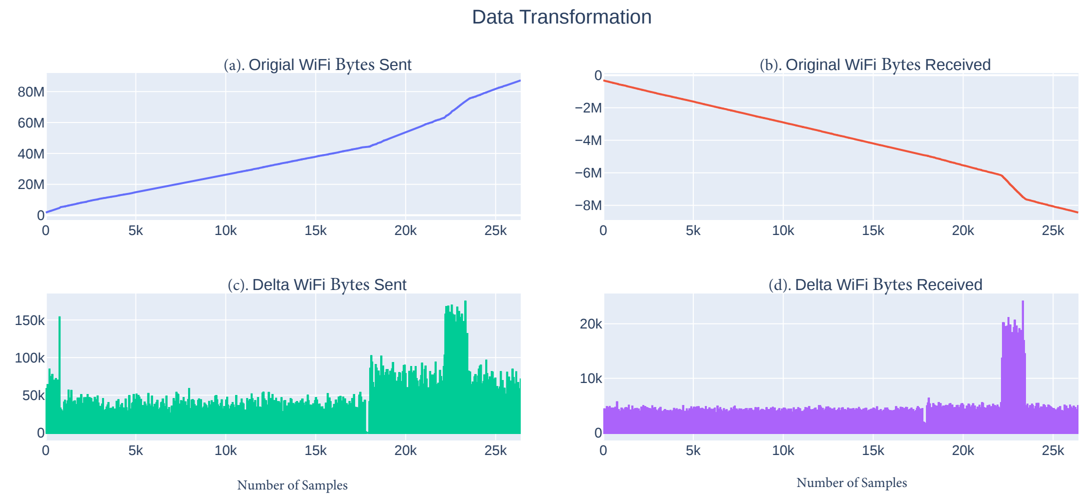

Figure 5.8: Transforming bytes sent $[x(i-1) - x(i)]$ and received $[x(i) - x(i-1)]$ over WiFi network into delta to eliminate trends in data.

An autoencoder was trained to detect anomalies, and a transformer-based model was used to classify them, both with the original variables and the variables adjusted to the delta, to validate the advantage of using the delta value. Table 5.4 summarizes the ability of models that use the original signals from bytes sent and received over WiFi to detect and classify anomalies versus the same models trained with the delta value of the original signals. The first case observes the poor performance of the detector and the classifier. Both have difficulty detecting attacks and some failures, reflected in the low F1 score. While the models trained with the delta signals present performances above 90%.

Table 5.4: Transforming the features related to the bytes sent and received over WiFi and using the delta of these variables improve the model's ability to detect anomalies.

Model	**Delta**	**F1**	**F2**	**F3**	**A1**	**A2**	**F1-Score**
Detector	Yes	✓	✓	✓	✓	✓	0.963900
	No	✓	✓	✗	✗	✗	0.542901
Classifier	Yes	✓	✓	✓	✓	✓	0.900422
	No	✓	✓	✗	✗	✓	0.650023

The previous paragraphs are visually explained in Figure 5.9. On the left side of the figure, you can observe the outcome of the autoencoder trained to detect anomalies using the delta of the variables of bytes sent and received over WiFi.

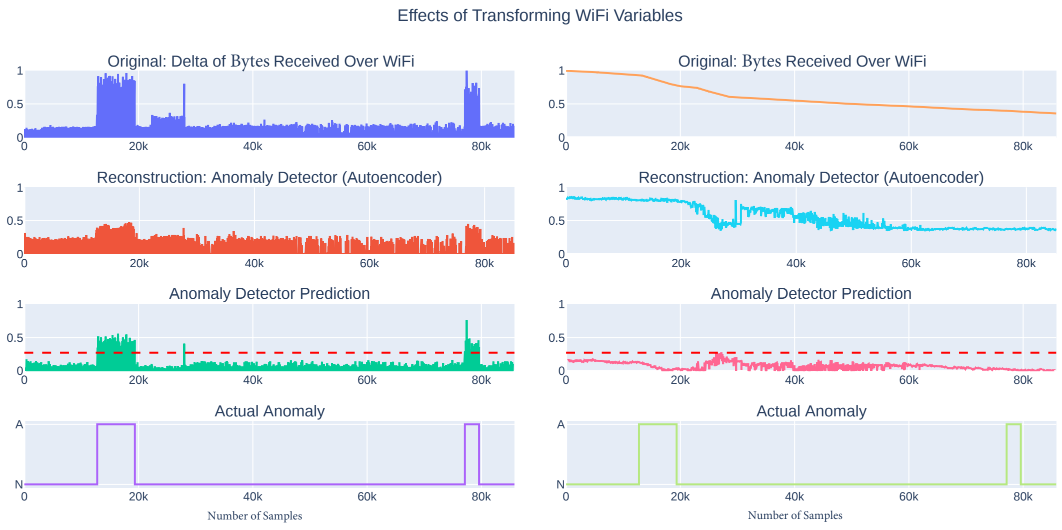

Figure 5.9: Transform of Features related to WiFi. On the right side, the autoencoder used as a detector does not reconstruct the signal. On the left, the autoencoder detects the anomalies correctly. The horizontal axis represents the number of samples.

The input of the autoencoder is represented above in blue; next, the reconstruction and the subtraction between the two previous plots are displayed. A dashed line indicates the threshold level. Any value above this threshold level is classified as *anomaly*. Compared with the actual anomaly (below), it can be concluded that this detector performs well. On the contrary, the result of a model trained with the original WiFi signals is shown on the right side. In this case, the autoencoder does not adequately reconstruct the input or recognize a pattern that can be labeled as an anomaly. From now on, the delta value of the bytes sent and received over WiFi will be used, and the original signals will be discarded.

5.3.5 Handling Outliers

In this work, outliers were handled on two occasions. One, outliers were eliminated from normal data before calculating the threshold values for the anomaly detector; this will be discussed in later sections. The other was before scaling the data between zero and one [0,1] to train machine learning models. In the latter, huge values were truncated to a value considered acceptable by the knowledge domain.

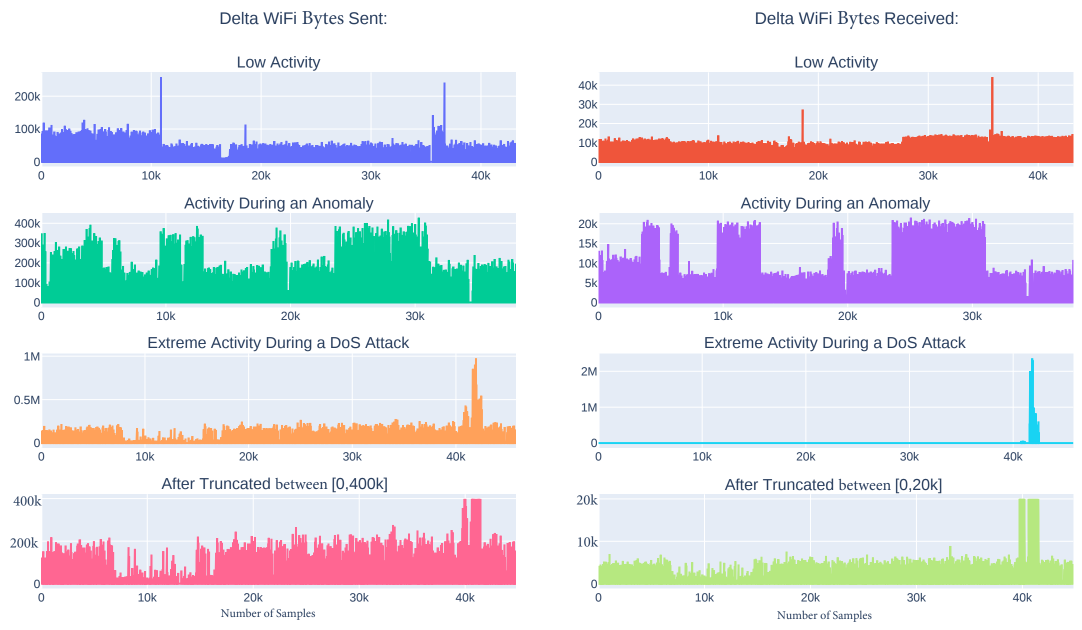

Figure 5.10: Handling Outliers. Extreme values in WiFi features have been truncated to ensure that the classifier still detects small variations evidenced by some anomalies after scaling the data.

Figure 5.10 illustrates this point. The plots in blue and red (above) indicate that when the network is experiencing low activity, the delta values for sent bytes remain below 100k bytes, and received bytes remain below 10k bytes, except for some outliers. However, during increased network activity, these delta values increase to 400k and 20k bytes, respectively, as shown in the second row of plots. In some instances, such as a DoS (denial of service) attack, these delta values can increase to over 1 Megabyte, as shown in the following plots. If we scale these features with such extreme values, the patterns that describe other anomalies will go unnoticed by the detector and the classifier. To avoid this, we have truncated these values to 400k and 20k bytes, respectively. This truncation does not pose any problem while putting the framework into production since we can consider this aspect while collecting the data. Finally, the effect of this truncation can be observed in the plots shown below.

5.3.6 Data Scaling

Datasets often contain features that have varying units and scales. These differences can result in a biased model that prioritizes larger-scale features. To avoid this issue, it is essential to normalize or standardize the data so that each feature contributes equally to the model's prediction. As mentioned in Chapter 4, normalization is used when assuming a Gaussian distribution, dealing with features at different scales and algorithms that are sensitive to variance, while min-max scaling is used for data bound to a specific range and for ease of interpretation.

Table 5.5: Scaling Data. Data is not a Gaussian distribution. Minimal-max scaling is better since mean-std generates data outside the [0,1] range. The table shows information from only one of several CSV files.

Scaling		**SoC Temp**	**Load1**	**WiFi Sent**	**Current**	**Sound**	**Freq**
Mean_Std	min	-1.385498	-1.931534	-1.16142	-1.330004	0	0
	max	0.978175	4.968823	-0.495499	5.899065	247	4000
Min-Max	min	0.105186	0.002449	0.61101	0.035836	0	0
	max	0.368395	0.165765	0.77309	0.635752	0.622	1

As noted in the data exploration section, the time series collected do not follow Gaussian distributions. This means that when we used the mean and standard deviation to scale the data, they were not scaled between zero and one. Therefore, in this case, we used the min-max technique. Table 5.5 shows the interval values for some variables after normalization and min-max scaling. In the first case, negative values and values greater than one were observed. In the second case, all data fell between 0 and 1. It is important to note that some data did not reach the extreme values 0 and 1 because other datasets had higher or lower values that influenced standardization.

5.3.7 Non-Numerical Features Encoding

Machine learning models are often designed to process numerical data. To achieve this, categorical data must be converted to a numerical format using one-hot encoding, label encoding, or embedding layers. In the case of an anomaly detector, a single-column output is used. The values in this column represent normal data (value=0) and anomalies (value=1). For the classifier, three columns were coded: normal (000), failure (010), and attack (100).

5.3.8 Sequence Creation

As discussed in Chapter 2, transforming time series data from $[n\_samples, n\_features]$ to $[n\_samples, n\_times, n\_features]$ enhances learning dynamics, efficient data utilization, flexible forecasting, feature utilization, model input requirements, and temporal dependency capture. Figure 5.11 graphically shows the process of making this transformation. Each sequence of n_times samples is pooled separately for each feature.

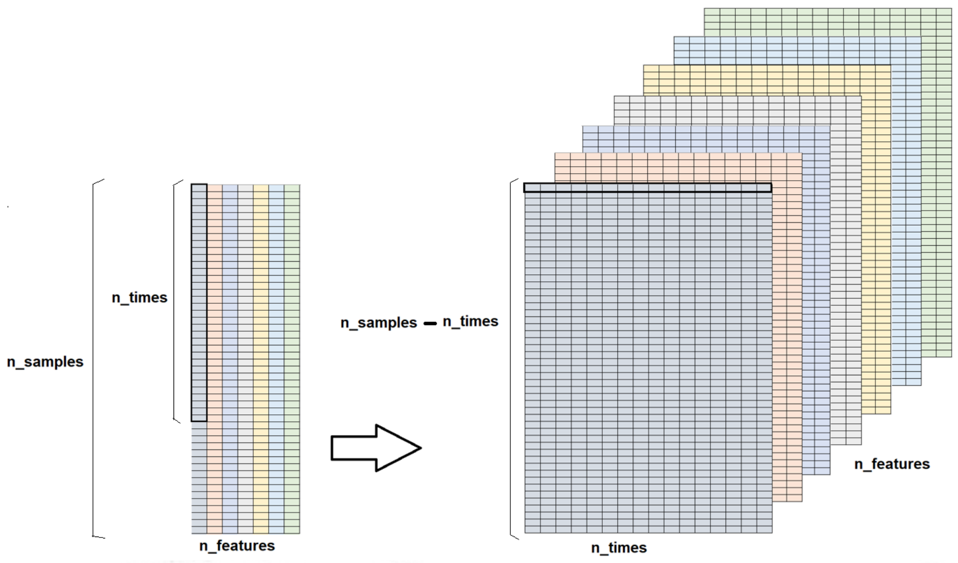

Figure 5.11: Graphical representation for transforming time series data from *[n_samples, n_features]* into *[n_samples, n_times, n_features]*

The figure shows that the new dataset takes up much more memory space. For further illustration, please note the n_times vertical samples highlighted in the figure on the left and their horizontal location on the right. The window size that slides through the data is essential to creating the sequence. The window size must be sufficient to detect patterns but not so extensive that it dilutes the identification of anomalies. Due to the absence of a universal solution, experimentation is essential to balance relevance and computational efficiency. Table 5.6 shows the F1 score for two classifiers, a Transformer and an LSTM model. According to this result, a window with 300 samples is the best option for the case study and will be used in later models.

Table 5.6: Length sequence data. Two anomaly classification models with different sliding window sizes were evaluated based on their f1 score, and a length of 300 samples was the best for these classifiers.

Window	**Transformer Model**	**LSTM Model**
100	0.799250	0.754377
200	0.812366	0.773365
300	**0.833983**	**0.781802**
400	0.804396	0.684335
500	0.778157	0.617783

5.4 Feature Selection

Specific machine learning algorithms can be computationally expensive and require a long time to train on datasets with many samples. To speed up the training process, a feature selection approach has been followed to reduce the number of input variables and simplify the data, making it easier to work with during the training phase. Statistical techniques and domain knowledge have been combined to discard variables that should not be part of the training set. Figure 5.12 shows a variable correlation analysis based on the Spearman index; it was detected that the CPU load in the last 5 minutes was strongly correlated with the load of 1 and 15 minutes. The same thing happened with the frequency and voltage of the CPU. As explained in the previous section, bytes sent and received over WiFi were replaced by their delta values. The temperature and relative humidity in the building where the CPS operates are not directly related to what happens in the IIoT and were considered to be discarded.

Finally, during the data exploration, it was observed that the CPU's available and free memory did not show significant variations when an anomaly occurred; therefore, they

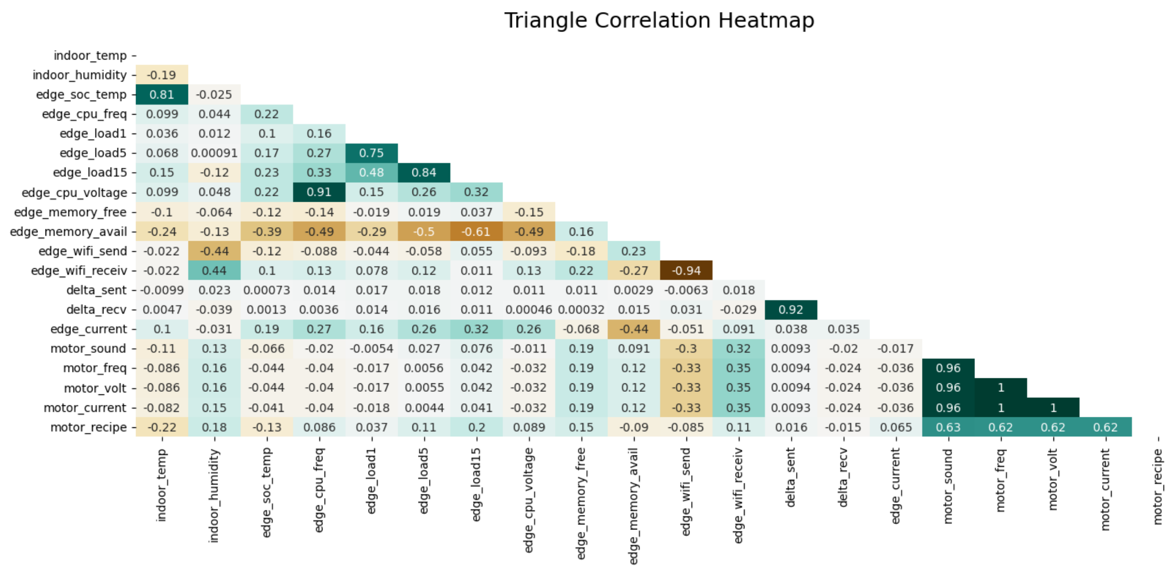

Figure 5.12: Correlation of Variables. In the training set, variables strongly correlated with each other should not be chosen since they provide little additional information to the model but represent a computational cost.

were identified as candidates to be discarded. An anomaly detection model was trained for 10 epochs, with and without each of the above-mentioned variables, to decide whether a variable should remain in the training set. Table 5.7 shows the metrics obtained for each case.

Table 5.7: Feature Selection. The anomaly detector model was trained with all IIoT and context variables. Then, variables were eliminated one by one, and the model was trained again until the training set was reduced to 8 features.

Features	**Excluded**	**Reason**	**Precision**	**Recall**	**F1 score**
15	wifi_sent, wifi_receiv	It was used delta_sent & delta_receiv	0.518628	0.327521	0.401446
13	indoor_temp, %humidity	Not related to IIoT or CPS	0.912609	0.522313	0.664335
10	cpu_freq, memo_free, memo_avail	Changes not related to anomalies	0.933241	0.529629	0.675710
8	load5, cpu_volt	Strongly correlated to other variables	0.922616	0.630521	0.749053

The best performance was obtained with a training set that included internal temperature, 1—and 15-minute load, the bytes sent and received delta, CPU current consumption, and motor sound. Motor frequency was also included as context information to improve model performance.

5.5 Context Information

The cyber-physical influences the IIoT dataset, which affects certain variables. For instance, the sound sensor is impacted by the operating frequency of the nearest motor. Figure 5.13 demonstrates that the frequency of the motor provides insight into the abnormal behavior of the sound signal. This is particularly noticeable during a data injection attack (on the left) or a sensor disconnection failure (on the right). If the sound level fluctuates while the motor frequency remains unchanged, this indicates the presence of an anomaly. Therefore, the frequency of the motors has been used as "context information" for the dataset extracted from the IIoT system to detect and classify anomalies.

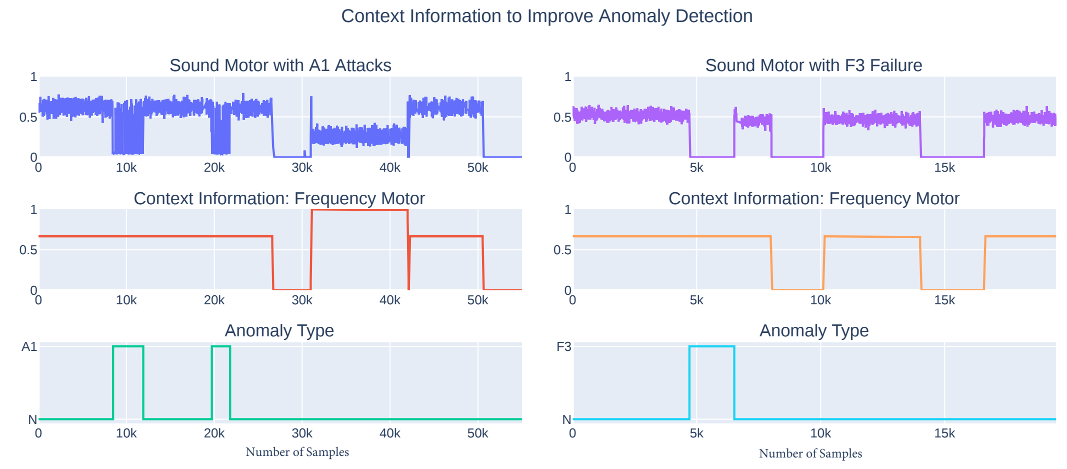

Figure 5.13: Context information. By incorporating motor data into the Industrial Internet of Things (IIoT) dataset, it becomes possible to distinguish between normal and abnormal behavior.

Table 5.8 summarizes the effect of context information on anomaly detection and classification models. When they do not use the CPS feature, none of the models recognize the sound sensor disconnection failure (F3), and the classifier does not recognize the data injection attack. The above reduces the F1-score in both models.

Table 5.8: Effect of context information on anomaly detection and classification models. Context information helps classify all anomalies.

Model	Context	F1	F2	F3	A1	A2	F1-Score
Detector	Yes	✓	✓	✓	✓	✓	0.963900
	No	✓	✓	✗	✓	✓	0.926210
Classifier	Yes	✓	✓	✓	✓	✓	0.900422
	No	✓	✓	✗	✗	✓	0.833197

The negative effect of the lack of context information on the anomaly detector and classifier is graphically observed in Figure 5.14. The plot above on the left shows that A1 attacks are classified as events, meaning that the classifier does not know the nature of the anomaly. Something similar occurs with fault F3, above right, which is also classified as an event. Meanwhile, models that incorporate context information mostly correctly detect and classify these failures and attacks (plots in the middle left and right). Below, on both sides, you can see the actual classification of each anomaly

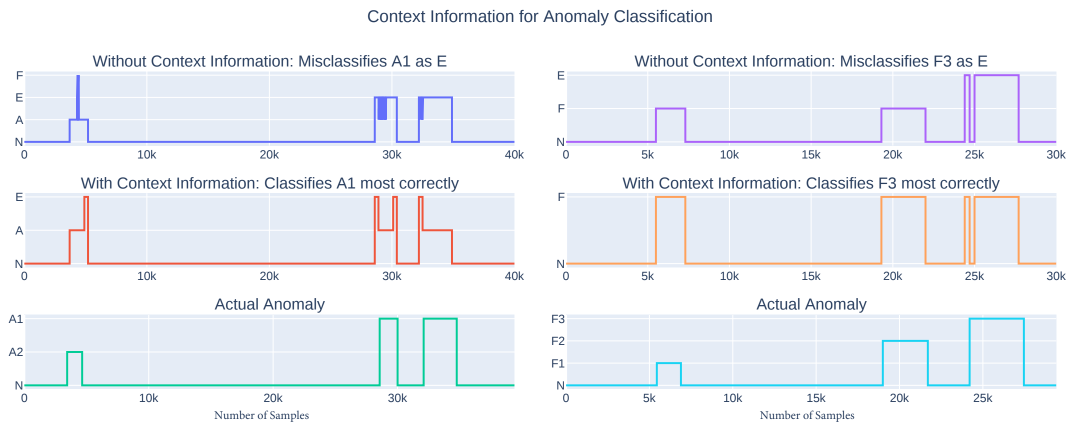

Figure 5.14: Context information helps classify correctly attacks (A1) and failures (F3). Without context information, A1 and F3 are classified as Event

5.6 Data Balancing

In scenarios where certain classes are not evenly represented, resampling the dataset can help balance the class distribution. This implementation involves undersampling the majority of the normal class. Three strategies have been tested to balance the training sets. In the first one, data trimming was done around each anomaly, taking a proportional amount of normal data on either side; the rest was discarded. The second strategy was to randomly eliminate as much normal data as necessary to balance the amount of all classes. The dataset was first balanced in the above methods, and then the sequence of the form $[n\_samples, n\_times, n\_features]$ described in previous sections was generated. In the third strategy, the data sequence was first created, and then the sequences necessary to balance the set were randomly removed. The latter guarantees that the sequence contains consecutive samples, which seems to facilitate the training of transformer or LSTM-type models, according to the results in Table 5.9, where the latter balancing strategy shows a higher F1-score.

Table 5.9: Strategies for Balancing Data. Two classifiers have been trained using balanced data with three strategies to undersample the normal class.

Balancing Strategy	**Metrics**	**Transformer**	**LSTM**
Trimming data around the anomaly	Loss	0.198894	0.313834
	Precision	0.799662	0.769618
	Recall	0.828263	0.665499
	F1-score	0.813661	0.713732
Undersampling majority class after creating sequences	Loss	0.105932	0.340602
	Precision	0.896060	0.805174
	Recall	0.794975	0.823306
	F1-score	**0.842446**	**0.814089**
Undersampling majority class before creating sequences	Loss	0.187673	0.111698
	Precision	0.817306	0.810257
	Recall	0.458463	0.773303
	F1-score	0.587372	0.791299

5.7 Anomaly Detection

The framework described in chapter 4 proposes using an anomaly detector followed by a classifier. Using two models instead of one aims to reduce the number of false positives and negatives while increasing the possibility of detecting zero-day anomalies, that is, those not available when training the models. The latter is achieved using unsupervised models that did not require labeled data during training, as with autoencoders.

5.7.1 Autoencoder

Autoencoders are a type of machine-learning model that is commonly used for detecting anomalies. These models work unsupervised and consist of two networks: an encoder and a decoder. The encoder processes the input data and converts it into a latent representation. Then, the decoder takes this representation and generates an output. By comparing the input and the output data, the differences can be used to identify anomalies in the original data [53]. In anomaly detection, an autoencoder learns to compress and decompress normal data efficiently. When an anomalous piece of data (which significantly differs from the norm) is fed into the autoencoder, the reconstruction error (the difference between the input and the reconstructed output) tends to be higher. Here, a threshold for the reconstruction error is set. The data point is classified as an anomaly if the error exceeds this threshold. This is described graphically in Figure 5.15; the first plot (above) shows the delta signal of the packets received over WiFi, and then it shows the autoencoder's reconstruction of this signal. The plot below shows the subtraction between both signals and the threshold value as a dotted line. Any value over this line is labeled as an anomaly. In this thesis, two autoencoders have been trained, and their performance was compared.

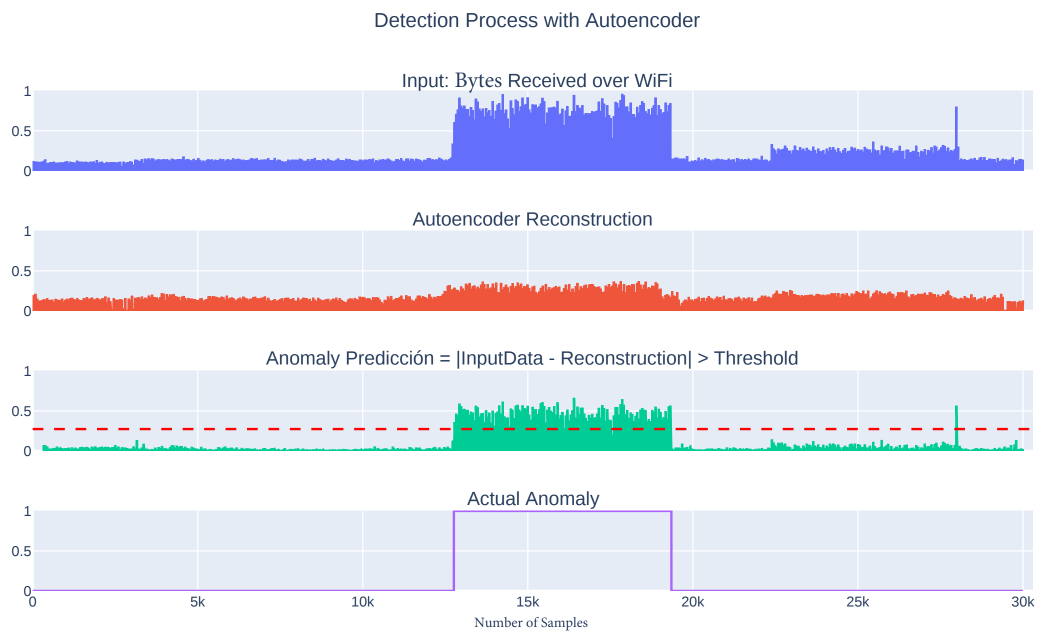

Figure 5.15: Anomaly Prediction with Autoencoder. Any sample above the threshold value is labeled as an anomaly.

In this framework implementation, one-dimensional convolutional (conv1D) alternating with dropout layers have been used for the encoder and transposed convolutional with dropout layers for the decoder. Conv1D is a type of convolutional layer designed explicitly for 1D data. It is commonly used in sequential data models, such as time series. This layer performs a convolution operation involving a sliding filter moving along the input data, effectively allowing the model to learn from local patterns within the sequence. A dropout layer is a simple and effective technique used in autoencoders to prevent overfitting. The dropout method randomly sets a fraction of input units to zero at each update. This randomness helps prevent the model from relying too heavily on any one or a few neurons. Moreover, transposed convolutional layers perform the reverse operation of a standard convolutional layer. While a standard convolution reduces the spatial dimensions of the input, a transposed convolutional layer increases the spatial dimensions of its input. Figure 5.16 shows the schematic of the anomaly detector. When the threshold value that monitors the reconstructed signal is exceeded, the anomaly classifier is invoked. Otherwise, the sample is labeled as normal.

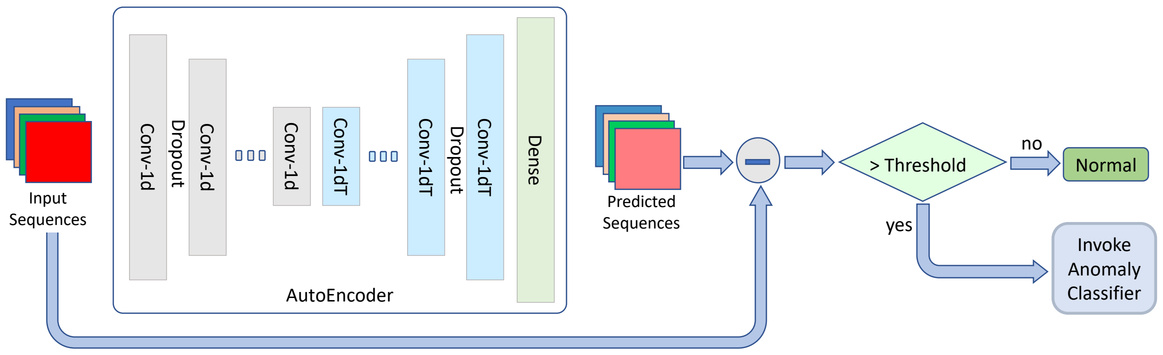

Figure 5.16: Autoencoder-based anomaly detector. The 1D-convolutional model reconstructs the input signal. The anomaly classifier is invoked if both signals' differences exceed the threshold value.

An additional autoencoder has been implemented using LSTM coupled with dropout layers for both the encoder and decoder. The output is wrapped with a TimeDistributed layer that applies a Dense layer to each time step of the input tensor separately. LSTMs are a type of recurrent neural network (RNN) ideal for handling sequence data, such as time series, due to their specialized architecture, which is particularly effective at capturing long-term dependencies in sequence data. The LSTM encoder is responsible for compressing the input sequence into a fixed-size context vector that represents the most essential features of the input data. At the same time, the decoder tries to reconstruct the original input sequence. The TimeDistributed layer wraps another layer, such as a Dense layer, and applies it to each time step of the input tensor independently. This implies that the same layer weights are used for each time step, but the operation is carried out separately for each slice. The following section shows a comparison between these two autoencoders.

5.7.2 Thresholds

The threshold value separates normal values (lower) from anomalies (higher). The strategies for its calculation were described in chapter 2. In this study case, the maximum difference value between the input and the reconstructed output (reconstruction error) is used as the threshold, as shown in Figure 5.17, where the dotted line representing the threshold value passes just through the highest value level. It should be noted that a threshold value is calculated for each feature.

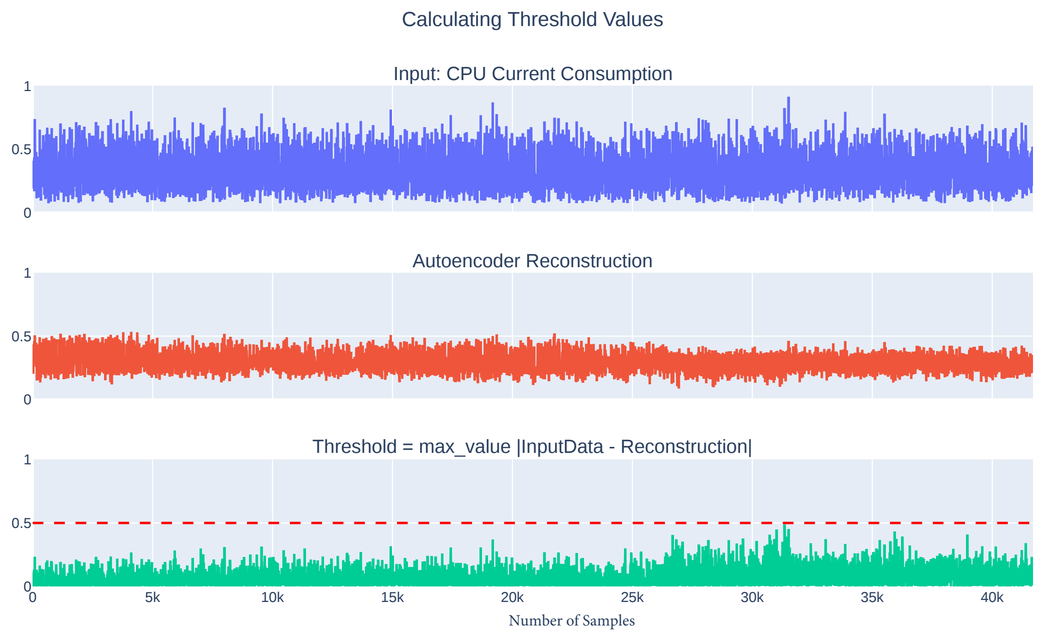

Figure 5.17: Threshold Value. The maximum value of the reconstruction error made by the autoencoder is chosen as the threshold (dashed line).

When calculating the threshold values, it must be considered that although datasets with normal data are used, they may present outliers not reconstructed by the autoencoder. This causes the reconstruction error (difference between the input and the reconstruction) to be higher, and therefore, so is the threshold value, affecting the performance of the autoencoder when it is deployed in production. To solve the above issue, the outliers from the normal data must be removed before calculating the threshold values. Figure 5.18 shows the effect of removing the outliers; the plot above shows extreme values every time the sound sensor abruptly changes level. To eliminate them, all values above mean plus three times standard deviation are replaced by the comparison value (if $value > mean() + 3 * std()$ then $value = mean() + 3 * std()$); the result is shown in the plot below, where the graphic appears without outliers. Since some signals, such as sound, change their reference level considerably, calculating the mean and standard deviation was carried out by ranges of values where the motor speed (context variable) remained constant. The outlier removal process was automated through a Python script.

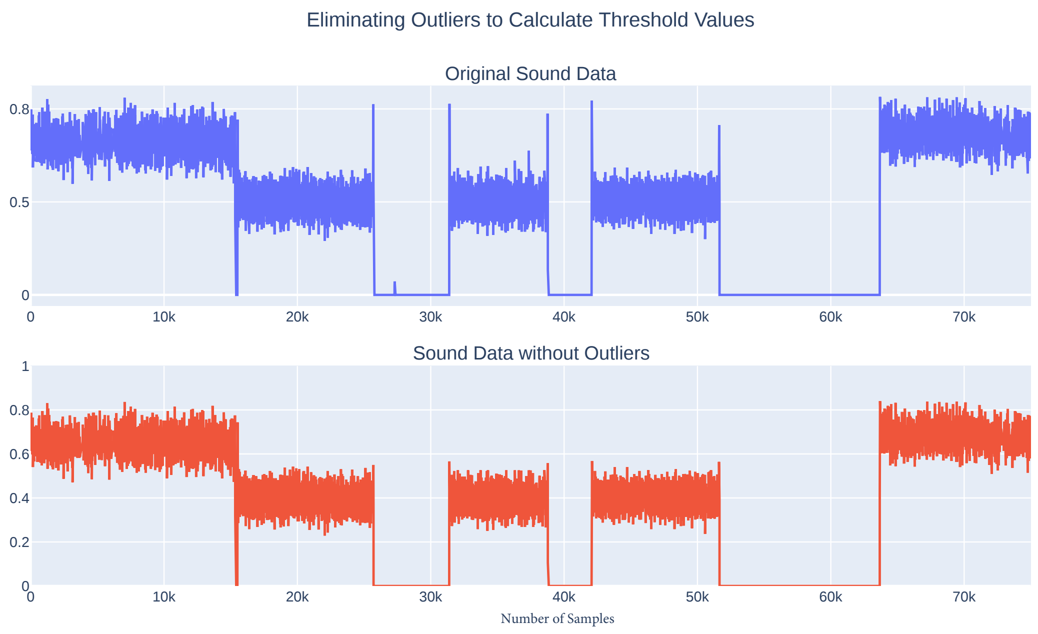

Figure 5.18: Removing outliers before calculating the threshold value. Outlier values (above) are replaced by $mean() + 3 * std()$. This operation is carried out by ranges of values where the behavior of the CPS remains constant to guarantee the average value of the signal to be pruned.

5.7.3 Metrics

Several metrics have been proposed in the literature to evaluate the performance of machine learning models. One of the most used is accuracy, which measures the number of samples correctly classified. However, in this study case, there are unbalanced datasets with much higher amounts of normal data than data labeled as anomalies. This generates high accuracy even though the model performance is very poor. Instead, precision, recall, and f1-score metrics that do not involve the number of true negatives (normal values) in their equations are recommended. The equations of these metrics are shown in Table 5.10. On one hand, Recall shows how many anomalies were not detected; many false negatives lead to a low recall. On the other hand, precision shows how much normal data was identified as anomalies; a high false positive rate produces a low precision. While F1-score provides an estimate between the false positive and false negative rates. Chapter 2 shows a more detailed description of these and other metrics.

Table 5.10: Evaluation Metrics for Anomaly Classification. For unbalanced datasets, it is advisable to use metrics that do not involve the majority class, such as precision, recall, and F1 score.

Metric	**Formula**	**What It Evaluates**
Accuracy	$\frac{TP+TN}{TP+TN+FP+FN}$	Overall correctness
Sensitivity (Recall)	$\frac{TP}{TP+FN}$	Correctly identified positives
Precision	$\frac{TP}{TP+FP}$	Correctness of positive predictions
F1-Score	$2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision}+\text{Recall}}$	Balance between precision and recall

5.8 Anomaly Classification

In anomaly classification, the objective is to predict a discrete label for a given input from a set of predefined classes. This is generally a supervised learning task where a model is trained on a dataset with known labels, learning to classify new data based on this training. In this case, two classifiers, a transformer-based and an LSTM-based model, were trained to predict whether the data is normal, an attack, or a failure. Initially designed for natural language processing tasks, a Transformer neural network can be used as an anomaly classifier for time series. The transformer model used here is similar to the original encoder proposed in [49]. The core component of Transformer models is the self-attention mechanism, which allows the model to weigh the importance of different parts of the input data relative to each other. For anomaly detection, this means Transformer can simultaneously consider the entire sequence of inputs, determining which features are most abnormal or indicative of an anomaly. Expanding on self-attention, multi-head attention facilitates joint attention across different representation subspaces. This is particularly useful in anomaly detection as it can help the model capture multiple types of relationships or anomalies in the data. Figure 5.19 shows the schematic of the transformer used in the anomaly classifier. We do not use the input embedding or positional encoder mechanisms; instead, the temporal sequences described in previous sections were delivered directly to the encoder block. The rest of the layers, feed-forward, and normalization were implemented as in the original version in [49].

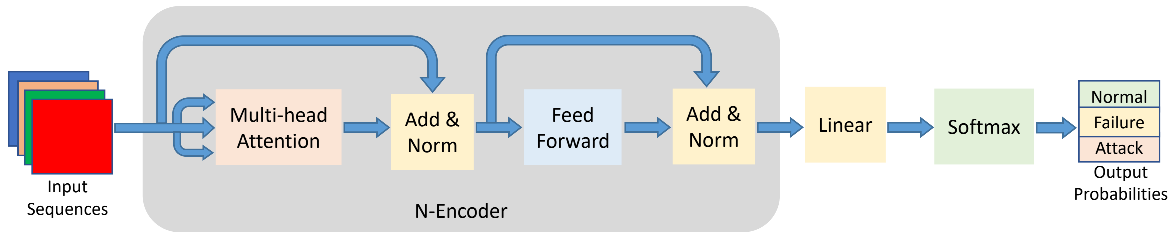

Figure 5.19: Transformer-based classifier. Data sequences are delivered directly to the encoder without input embedding or positional encoding layers.

The second anomaly classifier uses LSTM layers alternating with Dropout to reduce the risk of overfitting and a TimeDistributed wrapping of a dense layer. An LSTM network comprises a series of LSTM units, each responsible for maintaining a memory cell that can retain information across time steps. In the next section, the performance metrics of both classifiers are compared when the anomaly detector invokes them.

5.9 Anomaly Detector and Classifier Together

Figure 5.20 shows the schematic of the anomaly detector and classifier operating together. The detector implements an autoencoder model, which alerts of anomalies and invokes the classifier to determine whether it is a failure or an attack. If the classifier generates an ambiguous diagnosis indicating that the detector wrongly labeled the sample and that it is actually *normal*, an *event* label is generated so that the human operator can classify it manually. Two anomaly detection models were trained, one based on convolutional autoencoders and the other on LSTM autoencoders. Additionally, two classifiers were evaluated, one transformer-based and the other LSTM-based.

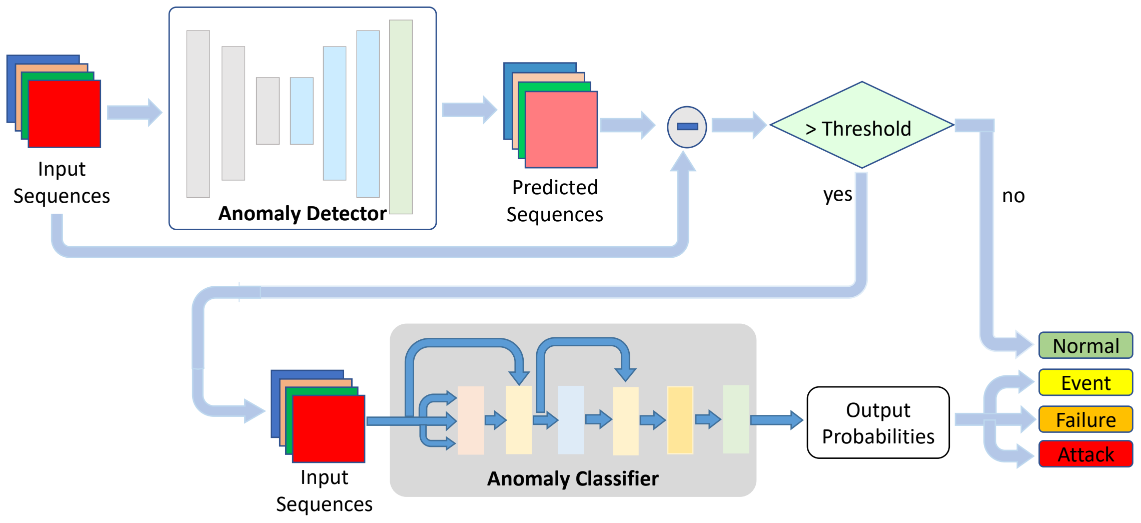

Figure 5.20: Anomaly Detection and classification. If the difference between the input and the reconstructed signal exceeds the threshold value, the classifier is invoked to classify the samples between failures and attacks; if it is classified as normal, it is labeled as an event so that the operator can classify it manually.

Table 5.11 shows the capacity of each model to avoid false negatives (recall) and false positives (precision) and the harmonic mean of both (F1). The convolutional-based autoencoder is the best detector, while the transformer-based classifier performs best in the three metrics. More detailed information about framework evaluation for anomaly classification in IIoT is presented in the next chapter.

Table 5.11: Comparison between anomaly detector and classifier models. Conv1d-based autoencoder and transformer-based classifier show better precision, recall, and F1 score metrics.

Block	**Model**	**Precision**	**Recall**	**F1-Score**
Detector	Conv1d	0.973610	0.913190	0.942433
	LSTM	0.962211	0.870235	0.913915
Classifier	Transformer	0.893578	0.944319	0.918198
	LSTM	0.880367	0.894792	0.887471

5.10 Discussion

The results presented in this chapter underscore the efficacy and potential of the proposed IIoT anomaly classification framework in identifying and categorizing anomalies within industrial IoT systems. The comprehensive experimental setup, involving an emulated Cyber-Physical System (CPS) and a robust Industrial Internet of Things (IIoT) monitoring system, provided a realistic testbed to evaluate the framework's performance. The framework demonstrated high detection accuracy and low false positive rates across various test scenarios, highlighting its reliability in real-world applications. The convolutional autoencoder-based anomaly detector outperformed the LSTM-based model in terms of precision, recall, and F1-score, indicating its superior capability to differentiate between normal and abnormal data patterns. This suggests that convolutional layers are more effective in capturing spatial dependencies within the data, which are critical for accurate anomaly detection in IIoT environments.

Integrating advanced machine learning algorithms and real-time data processing techniques enabled the framework to promptly and accurately identify anomalies, reducing potential downtime and enhancing operational safety. The comparative analyses demonstrated the framework's robustness and adaptability, making it a viable solution for various industrial applications. The case studies and real-world scenarios further validated the framework's practical applicability. The successful identification of failures and cyber attacks within the testbed highlights the framework's comprehensive approach to anomaly classification. By leveraging data from multiple sources and employing classification models, the framework provides actionable insights that can facilitate timely interventions and mitigate the impact of anomalies on industrial operations.

Despite the promising results, several challenges and limitations were encountered during the implementation. Data quality and noise variability posed significant challenges in maintaining high detection accuracy. Additionally, the framework's performance heavily relied on the quality of data preprocessing and feature engineering, underscoring the importance of these stages in the anomaly detection pipeline. The framework's scalability in larger, more complex industrial environments also requires further investigation. Looking ahead, several avenues for future research and improvement have been identified. Enhancing the framework's ability to handle diverse and noisy data through advanced preprocessing techniques and robust feature selection methods could improve its performance. Exploring the integration of additional data sources, such as environmental and contextual information, could provide a more holistic view of the operational environment, enhancing anomaly detection accuracy. Additionally, real-time adaptation and learning capabilities could be incorporated to enable the framework to evolve with changing operational conditions and emerging anomaly patterns.

In conclusion, the proposed IIoT anomaly classification framework has demonstrated substantial potential to enhance the reliability and safety of industrial systems. Its high detection accuracy and practical applicability make it a valuable tool for modern industrial environments. By addressing the identified challenges and exploring future research directions, the framework can be further refined and adapted to meet the evolving needs of the industrial sector. The next chapter presents other results related to evaluating the performance of the optimized anomaly detection and classification models. Likewise, the challenges and opportunities for improvement of the framework are discussed.

Chapter 6

Results and Discussion

In this chapter, we build upon the previous one, in which we introduced the results of implementing the framework for classifying anomalies in IIoT. Firstly, we describe preliminary experimentation presented at the International Engineering Congress IC-EXPOI 2022, where we differentiated between a temperature failure, a denial-of-service attack, and the normal operation of an IIoT edge device. Then, this chapter describes the dataset used in evaluating machine learning models, which contains labeled failures and attacks. Next, we present the results of implementing the IIoT anomaly detection strategy, including the validation process to eliminate isolated samples that may be false positives; this is achieved by invoking the anomaly classifier only if the number of samples labeled by the detector in the sequence of 300 samples is greater than 15. We then discuss the cross-validation results used to evaluate the performance of the anomaly classifier. During the evaluation, we also used a dataset with unknown anomalies to test the model's ability to classify unseen anomalies. Moreover, lightweight versions of the models were run on a Raspberry PI3 to evaluate the ability of an edge device to run the anomaly classifier. Finally, we review the results, discuss any limitations of the implemented framework, and suggest potential improvements.

6.1 Preliminary Experimentation

In previous work, presented at the International Engineering Congress IC-EXPOI 2022 and published in [19], we explored the topic of anomaly classification in IIoT systems. The experimental purpose was to determine if it could differentiate between a temperature failure, a denegation-of-service attack, and normal operation of an Industrial Internet of Things edge device. An IIoT system was selected to simulate events, failures, and attacks in its edge device. This system consisted of several low-cost sensors that monitor a three-phase motor. Each sensor sends the motor current, temperature, sounds, speed, or vibration data through a microcontroller (ESP32) low-cost development system using MQTT to a Raspberry PI edge device. Raspberry processes motor data and activates alarms when a value exceeds a normal range. Then, the processed information was sent to a cloud application that stores and displays statistics on a website. An Edge device was monitored to detect some electronic component failures or cyberattacks. To differentiate between high-temperature failure or denial of service (DoS) attacks on an edge device processor, the Raspberry current consumption and some edge device internal data such as temperature, CPU load, frequency, voltage, and free memory were selected. The Raspberry PI collected the data. A fan regulated the edge device processor temperature. Moreover, the DoS attack was controlled by a computer running Kali Linux in a local network. This work found it possible to differentiate a temperature failure, a denial-of-service attack on the MQTT broker, and normal data using low-cost solutions. In that case, a current sensor and the internal data of the microcontroller were used. The analysis of variance (ANOVA) showed that failure, attack, and normal were statistically different. It also found that while current, frequency, and voltage differentiate between normal and attack, temperature allows differentiation between failure and normal system operation. The following section describes the results obtained by classifying anomalies in the proposed framework.

6.2 Case Study

As mentioned in the previous chapter, a testbed was established to evaluate the efficiency of the Anomaly Classifier Framework. The testbed included an emulated Cyber-Physical System (CPS) and an Industrial Internet of Things (IIoT) system. The CPS utilized in the testbed was an emulation of a real-life four-belt conveyor system, and the IIoT system monitored the CPS via a cloud platform, allowing for real-time data analysis and reporting. In addition to the results presented in Chapter 5, others of equal relevance within this thesis are presented here. First, a significant contribution of this thesis is the creation and publication of an IIoT dataset with anomalies labeled as faults and attacks. Second, implementation and evaluation of the framework for anomaly classification, using cross-validation with the training set, with a dataset that contains new anomalies not seen during training, and evaluation of lightweight versions of the models running on an edge device.

6.2.1 Dataset

Generating enough data is key for detecting and classifying anomalies, including failures, cyberattacks, and perturbations [191]. Some authors develop datasets using simulators [22, 53, 79, 191], testbeds [23, 46, 47, 50, 60], or online data [2, 9–11, 23, 35, 37, 45, 52, 55, 59]. Utilizing accessible online datasets can help cut down on collection expenses, and there are references available from researchers who have already utilized them. Nevertheless, these datasets have already established features and anomalies, making the introduction of new failures or attacks challenging. In the early phases of this thesis, an exhaustive search was performed on online datasets taken from IIoT systems with anomalies labeled as failures and attacks. However, the search was unsuccessful. We did not find a dataset that separately labeled enough failures and attacks, so we used a testbed to build a set with data that met the established requirements. This dataset is available online in a public repository [192].

We collected data for 10 days, storing information for approximately 4 hours daily. The data has been stored in 48 files, of which 31 contain normal data, 13 contain failures due to high temperature (F1), miscalibration (F2), and disconnection (F3), and 4 files contain attacks type DoS (A2) and data injection (A1). Each file can contain one or more anomalies of the same or different types. There are 2105134 normal records, 60463 records labeled as failures, and 38755 as attacks. Table 6.1 summarizes these data. Files with F3 failure were not used to train the model but to evaluate its ability to detect and classify anomalies not seen during training. This dataset will help address the shortage of ensembles for training models that classify anomalies into failures and attack categories. Likewise, it could classify specific types of failures or attacks.

Table 6.1: Dataset. The data was stored in 48 files, 31 with normal data, 13 with failures, and 4 with attacks. The anomaly types are, F1: High Temperature, F2: Miscalibration, F3: Disconnection, A1: Data Injection Attack, A2: DoS Attack.

Files	**Normal**	**Failures**	**Attacks**	**F1**	**F2**	**F3**	**A1**	**A2**
31	1254183	0	0	0	0	0	0	0
13	345547	60463	0	11	8	2	0	0
4	505404	0	38755	0	0	0	8	5
48	**2105134**	**60463**	**38755**	**11**	**8**	**2**	**8**	**5**

6.2.2 IIoT Anomaly Detection Strategy

In the anomaly detection stage, we chose an unsupervised model to increase the chances of detecting anomalies not part of the model's training data. Specifically, we trained several autoencoders, some with LSTM and others with CONV1D layers. The ones with CONV1D layers proved more effective in detecting anomalies in the available dataset, as shown in Table 6.2.

Table 6.2: Anomaly Detector and Classifier Architectures. Conv1d-based autoencoder and transformer-based classifier show better precision, recall, and F1 score metrics.

Block	Model	Precision	Recall	F1-Score
Detector	Conv1d	**0.973610**	**0.913190**	**0.942433**
	LSTM	0.962211	0.870235	0.913915
Classifier	Transformer	**0.893578**	**0.944319**	**0.918198**
	LSTM	0.880367	0.894792	0.887471

This section describes the optimized model for anomaly detection. First, the metrics used in this thesis to evaluate models are precision, recall, and F1-score. This is because none of these metrics involve true negatives TN, which are many and affect the balance of the calculations. Precision offers an idea of the false positives (FP) detected by each model; FP generates alerts for anomalies that do not exist, which unnecessarily occupies the attention of the human operator of the system. Recall reports false negatives (FN); these are abnormal samples overlooked and labeled as normal. Finally, F1 score is the harmonic mean between precision and recall. Table 6.3 shows these metrics for different combinations of the number of neurons in the latent space and the outer layer. The selected detector model is an autoencoder with 64-32-7 Conv1d layers in the encoder and 7-32-64 Conv1dT layers in the decoder. All layers alternate with dropout layers and use the Relu activation function, except for the output dense layer, which uses Sigmoid. The model was also compiled with optimizer Adam, a learning rate equal to 10e-4, and mean square error (MSE) as a loss function.

Sliding Window to Detect Anomalies

Some anomalies are not detected sustainably. Instead, samples labeled as anomalies alternate with more samples labeled as normal. Figure 6.1 shows this situation, from which we could affirm a priori that the anomaly is correctly detected. However, if we calculate the confusion matrix, the detector performs poorly.

Table 6.3: Optimization of detector model. Different combinations of neurons were evaluated in the latent space and the outer layer. Metrics in the table are precision (P), recall (R), and F1-score

Neurons in Latent Space	Neurons in Last Layer			
	64	128	256	512

4	P=0.808084
	R=0.951924
	F1=0.87412
		P=0.926982
		R=0.923562
		F1=0.92526
			P=0.881727
			R=0.919158
			F1=0.90005
				P=0.910769
				R=0.935526
				F1=0.92298

5	P=0.974133
	R=0.852114
	F1=0.90904
		P=0.970842
		R=0.906076
		F1=0.93734
			P=0.941134
			R=0.927826
			F1=0.93443
				P=0.943897
				R=0.930083
				F1=0.93693

6	P=0.999988
	R=0.868602
	F1=0.92967
		P=0.966125
		R=0.905189
		F1=0.93466
			P=0.965977
			R=0.930577
			F1=0.94794
				P=0.580565
				R=0.940041
				F1=0.71781

7	P=0.999424
	R=0.909402
	F1=0.95229
		P=0.988256
		R=0.866788
		F1=0.92354
			P=0.926175
			R=0.902689
			F1=0.91428
				P=0.867233
				R=0.943609
				F1=0.90381

The strategy used in the detector and the classifier is to take advantage of the sequence of 300 samples with which the model works. If, in the output sequence, there are at least fifteen samples that exceed the threshold value, the sequence is labeled as an anomaly. As a consequence of this algorithm, the anomaly prediction appears as a continuous line in the middle plot of the figure we refer to.

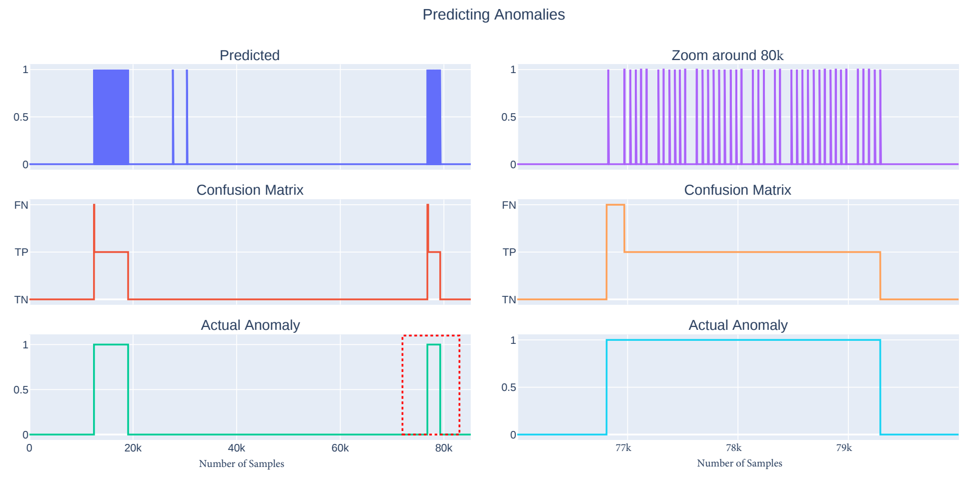

Figure 6.1: Some anomalies are detected intermittently, alternating with normal samples. Here is a zoom of around 80k on the right. If at least fifteen samples exceed the threshold value in the output sequence, it is labeled as an anomaly. Here, ‘1’ means anomaly, and ‘0’ is normal.

Table 6.4 shows the metrics calculated using the 300-sample sequence and the original anomaly points. The results indicate that the window is necessary for metrics to reflect the model's performance. Another consequence of this detection strategy is that outliers from a single sample are ignored and do not trigger the anomaly alert. The anomaly alert is only activated with 15 or more samples identified as such. It means that *anomalies* lasting less than three seconds are ignored because the sampling rate is 0.2s.

Table 6.4: Anomalies detected intermittently affect the metrics related to the confusion matrix (TP, TN, FP, FN) and, therefore, precision, recall, and F1-score. The sequence of the last 300 samples is used to validate if at least 15 samples have been labeled as anomalies to label the new sample.

Prediction	**TN**	**TP**	**FN**	**FP**	**Accuracy**	**Precision**	**Recall**	**F1-Score**
300-Sample Sequence	76119	8839	231	0	0.997288	1	0.974531	0.987101
Counting anomaly points	76113	518	8552	6	0.899541	0.988549	0.057113	0.107984

6.2.3 Anomaly Classification Results

The framework implemented in Chapter 5 proposes a two-stage classification model. First, an autoencoder evaluates a sequence with the latest 300 samples; if predicted as an anomaly, the classifier is invoked to determine whether it is a fault or an attack. However, the classifier may indicate that the sample is normal. In this case, the model labels it as an *event* so the human operator of the system can investigate it and take further action.

Several deep learning models based on LSTM and Transformer layers were trained to choose the best option for the multi-class classifier. Table 6.2 showed some metrics that helped choose the transformer-based architecture. In the classifier optimization process, models with different numbers of encoders and head attentions were evaluated. Table 6.5 shows the average metrics that helped choose an architecture similar to the original encoder introduced in [49] with 3 encoders and 20 head attentions. The model was also compiled with optimizer Adam, a learning rate equal to 10e-4, and categorical cross-entropy as a loss function.

Table 6.5: Optimization of the classifier model. Several models were trained by varying the number of encoders and head attentions. Metrics are precision (P), recall (R), and F1-Score

Multi	Encoders		
Head	2	3	4
Attention	P=0.887275	P=0.886532	P=0.884081
5	R=0.904949	R=0.926697	R=0.885009
	F1=0.89597	F1=0.90612	F1=0.88449
	P=0.867476	P=0.889589	P=0.870347
10	R=0.859819	R=0.922560	R=0.878698
	F1=0.86358	F1=0.90572	F1=0.87445
	P=0.883482	P=0.883524	P=0.868162
15	R=0.890425	R=0.878886	R=0.798919
	F1=0.88689	F1=0.88114	F1=0.83205
	P=0.852745	P=0.887309	P=0.880627
20	R=0.730866	R=0.927279	R=0.868781
	F1=0.78706	F1=0.90680	F1=0.87461

6.2.4 IIoT Anomaly Classification Model Validation

To ensure that the model performs well on new data, we utilized our version of the cross-validation technique to evaluate the classifier model. This involves splitting the data into subsets, training the model on some sets, and validating it on the remaining ones. Usually, the dataset is divided into k equally sized subsets, or folds, with a standard choice being k=10, known as 10-fold cross-validation. However, our data is stored in 15 CSV files of different sizes, each containing one or more failures or attacks. So, each of these files represents a fold. The model is trained on k1 folds and validated on the remaining fold for each. This process is repeated k times, with each fold used exactly once as the validation data. Performance metrics (precision, recall, F1-score) are calculated for each iteration. The final performance estimate is obtained by averaging these metrics across all folds. The results of this cross-validation are shown in Table 6.6. The average result provides a precision of 0.952, indicating a very low False Positive (FP) rate, typically related to remaining effects after the anomaly's cause has ceased; recall is 0.878, corresponding to a higher rate of False Negatives (FN) than FP. FN usually occurs right at the beginning of the anomaly due to a delay in detection or the middle when some samples are erroneously labeled as normal. Finally, the average F1-score is 0.91 for the evaluation set. When calculating precision, recall, and F1-score separately, we obtain 0.945, 0.797, and 0.862 for attacks and 0.954, 0.908, and 0.927 for failures, inferring that our system detects failures better than attacks.

In Figure 6.2, you can see the graphical representation of the classification results

Table 6.6: Cross-Validation for IIoT Anomaly Classifier.

File	Anomaly	Precision	Recall	F1-Score
edge1_11.csv	F1, F2	0.96	0.82	0.89
edge1_12.csv	F1, F1	1	0.8	0.89
edge1_13.csv	F2, F2	0.81	0.92	0.86
edge1_14.csv	A2	0.9	0.74	0.81
edge2_9.csv	F2, F1	0.89	0.97	0.93
edge2_10.csv	F1, F1	0.99	0.75	0.85
edge2_11.csv	F2	1	1	1
edge2_12.csv	A2, A1, A1, A1, A1	0.91	0.88	0.89
edge3_8.csv	F1, F2	0.97	0.99	0.98
edge3_9.csv	F1, F1	1	0.81	0.9
edge3_10.csv	F2	1	1	1
edge3_12.csv	A2, A1, A1, A1, A1	0.97	0.85	0.91
edge4_10.csv	F1	0.88	1	0.94
edge4_11.csv	F1, F2	1	0.93	0.96
edge4_12.csv	A2, A2	1	0.72	0.84
Total:		0.95	0.88	0.91

during cross-validation for the edge1 device. The graphs for the other edge devices can be found in Appendix B. All anomalies were identified, with a few datasets showing a small number of samples labeled as events. In the edge1_13 dataset, the number of samples incorrectly labeled as events slightly increased. After validating 15 samples (3 seconds) in the input sequence, the classifier in the middle eliminates false positives not associated with real anomalies detected by the detector on the left. The plots on the right display the actual anomalies.

6.2.5 Model Validation with New Anomalies

In the preceding section, we thoroughly analyzed the cross-validation results. The proposed system successfully identified all anomalies and did not produce any false positives that triggered alarms for non-existent anomalies. This upcoming section will examine the classifier's performance to uncover anomalies not part of the training set. We utilized two files containing sensor disconnection failures (F3) to evaluate the model. The results, as illustrated in Figure 6.3, indicate that the anomalies were indeed detected, but they were classified as normal events by the classifier. Consequently, the system activated an alert for the human operator to classify these anomalies manually. While this outcome is not ideal, it is nonetheless noteworthy that the system successfully detected the occurrence of an anomaly.

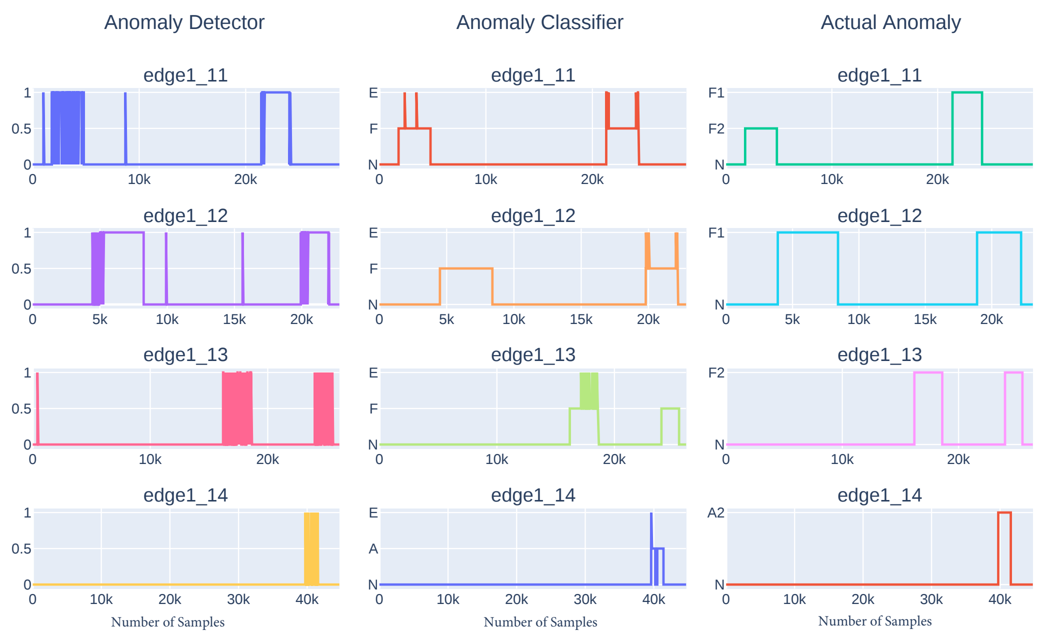

Figure 6.2: Graphical result for cross-validation with edge1 device data. Here '1' represents 'anomaly' and '0' represents 'normal'. Letters represent A: attack, E: event, F: failure, F1: temperature failure, F2: misconfigured failure, A2: DoS attack.

6.2.6 Deploying the IIoT Anomaly Classifier in a Testbed

To ensure timely anomaly detection, it is recommended that the classifier be run on each edge device. The deployment results are depicted here. The TFLiteConverter function set from the TensorFlow library generated a lightweight version of the classifier and detector models to run on a Raspberry PI3. Validation is performed with data collected at the same time that the training data was collected. The goal here is to evaluate the performance of the models when running a lightweight version of them on an edge device. The code used on the Raspberry to invoke the models appears in the appendix B. Figure 6.4 shows the four anomalies evaluated. There are no false positives that generate unnecessary alerts in the graph. In most cases, some samples are classified as events, meaning the detector classified them as anomalies, but the classifier labeled them as normal; however, they are very few compared to the correctly classified data, so the human operator does not doubt whether it is a failure or an attack. The classification graph of the dataset called "edge2_F1" shows a delay in detection; it starts around sample 3k, but the original anomaly (on the right) begins near 2k.

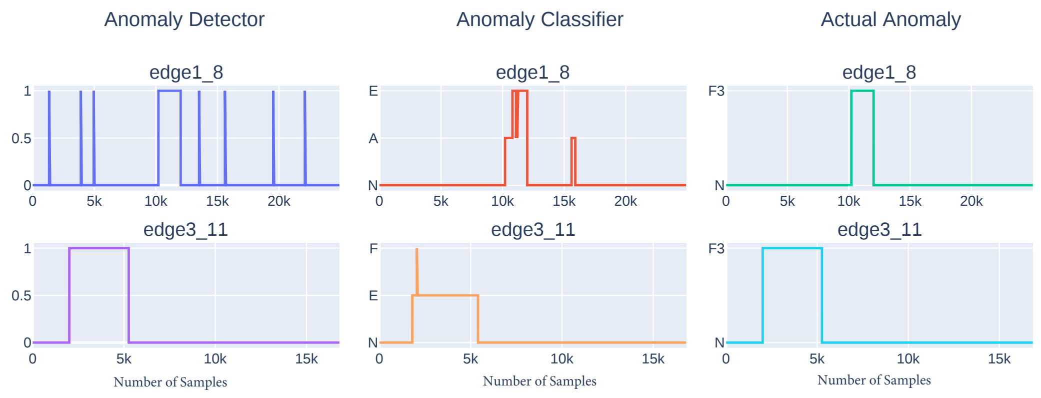

Figure 6.3: Model validation with new data. F3 anomalies were detected but classified as events.

The above is consistent with what was observed throughout the entire experimental process with the temperature failures (F1). Table 6.7 The table shows precision, recall, and F1-score values consistent with those obtained using the original model in Google Collaboratory, where the system shows slightly better performance classifying faults than attacks.

Table 6.7: Evaluating the IIoT Anomaly Classifier in a Testbed

File	Anomaly	Precision	Recall	F1-score
edge2_F2	Miscalibration Failure	0.88	0.97	0.92
edge2_F1	Temperature Failure	1	0.84	0.91
edge2_A2	DoS Attack	0.86	0.93	0.89
edge2_A1	Data Injection	0.91	0.88	0.89
Average		0.91	0.90	0.90

Table 6.8 summarizes the most important results of this work. The precision, recall, and F1-score metrics for the detector and both, with cross-validation and running on the testbed.

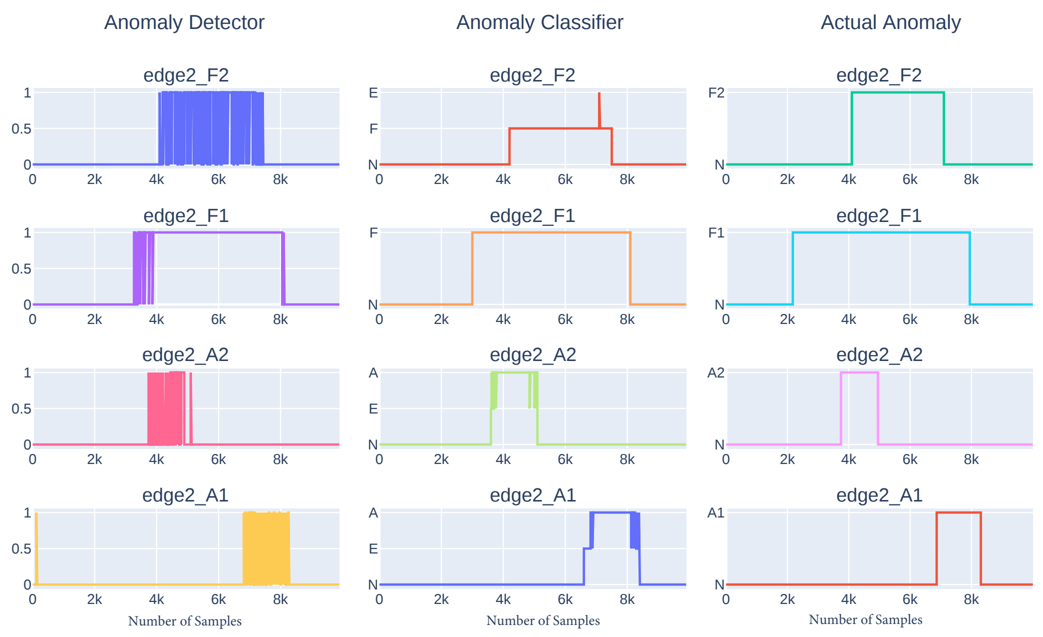

Figure 6.4: Evaluating the IIoT Anomaly Classifier in the Testbed. Detector labels samples as normal (0) or anomaly (1), and classifier labels as normal (N), event (E), failure (F), and attack (A).

Table 6.8: Summary of main results of the work.

Model	Precision	Recall	F1-Score
Detector	0.97	0.91	0.94
Both. Cross Validation	0.95	0.88	0.91
Both. Testbed	0.91	0.90	0.90

6.3 Discussion

The case study results demonstrate the effectiveness and robustness of the Anomaly Classifier Framework in detecting and classifying anomalies within an Industrial Internet of Things (IIoT) system. The case study's key contributions include creating a comprehensive IIoT dataset with labeled anomalies. The dataset generated for this study fills a significant gap in available IIoT datasets, which often lack the diversity and specificity needed for comprehensive anomaly detection and classification. By using a testbed that emulates a real-life four-belt conveyor system, we were able to produce a dataset with detailed labels for various types of anomalies, including high temperature

(F1), miscalibration (F2), disconnection (F3), Denial of Service (DoS) attacks (A2), and data injection attacks (A1).

The unsupervised anomaly detection model architecture (64-32-7 Conv1D layers in the encoder and 7-32-64 Conv1DT layers in the decoder) provided the best balance of precision, recall, and F1-score. This model's optimization, using metrics that do not involve true negatives (TN), ensures a realistic evaluation of its effectiveness in practical scenarios where false positives (FP) and false negatives (FN) are critical considerations. Using a sliding window technique to validate anomaly detection significantly improved the model's performance. The model reduces the likelihood of false alarms triggered by transient outliers by requiring at least 15 consecutive samples to exceed the threshold value before labeling an anomaly. This strategy aligns with the operational requirements of real-world CPS and IIoT systems, where stability and reliability are paramount. The two-stage classification approach, involving an initial detection by the autoencoder followed by a detailed classification using a transformer-based model, proved effective. With its optimized architecture (3 encoders and 20 head attentions), the transformer-based classifier achieved high precision and recall, particularly distinguishing between failures and attacks. The cross-validation results, with an average precision of 0.952 and an F1-score of 0.91, indicate that the model is reliable and can generalize well to new data. One notable aspect of the study was the model's ability to detect but not classify unseen anomalies (e.g., sensor disconnection failures, F3). These anomalies were identified but labeled as normal events, triggering an alert for human intervention. This outcome highlights the model's sensitivity and the importance of having a fallback mechanism where human operators can review and classify new types of anomalies. This feature is crucial for maintaining the system's adaptability and continuous improvement.

The results of this thesis have several implications for developing and deploying anomaly detection systems in IIoT. Creating a robust, publicly available dataset enhances the research community's ability to train and test new models. The demonstrated effectiveness of the Conv1D autoencoder and transformer-based classifier provides a strong foundation for future improvements and adaptations. Future work could further refine the model to improve its ability to classify previously unseen anomalies automatically. Additionally, exploring the integration of semi-supervised learning techniques could enhance the system's adaptability and reduce the reliance on human intervention. Finally, expanding the dataset with more varied anomalies and operational scenarios will continue to enhance the model's robustness and applicability in diverse real-world settings. In conclusion, the Anomaly Classifier Framework presented in this thesis significantly advances IIoT anomaly detection and classification. Its comprehensive dataset, optimized detection and classification models, and effective anomaly-handling strategies provide a robust solution for maintaining the safety and reliability of IIoT systems.

6.4 Challenges and Limitations

This section discusses the challenges encountered during the implementation and evaluation phases and how they were overcome. It also discusses limitations encountered during the investigation, such as limitations due to the dataset's nature or the model's computational complexity. The assumptions made during the research and how they may affect the generalization of the results are also mentioned.

- Resampling has been applied to integrate all signals into a single dataset. This has caused some variables to lose many samples that would have helped in anomaly detection and classification. An alternative would be to feed the models with the timing signals and their original sampling rates without integrating them into a single dataset.
- Although the models' performance improved due to the context information from the CPS, there is room for exploring other forms of context. For instance, domain-specific knowledge with specific rules could be utilized in the form of industrial process data. Alternatively, formulas involving mathematics or physical laws could create artificial data.
- Several hyperparameters were optimized to improve the performance of the models. This optimization process considered the time and computation available. It is possible that more precise tuning would have improved the models' metrics a little more.
- To balance the data, undersampling was used for the majority class representing normal data. Oversampling methods have yet to be explored, including GANs or simulators to generate new data with anomalies.
- The anomaly classification framework in IIoT was implemented and evaluated in a testbed. However, applying this framework to real industrial systems is necessary to identify improvement opportunities.

- A possible improvement in the framework would be to add a phase that provides specific instructions to implement cyber security measures such as identification, authentication, and authorization to guarantee the privacy, availability, and integrity of the data in the IIoT framework.

- The process of selecting variables was carried out using statistical analysis, domain knowledge, and recursive elimination of variables. Feature extraction techniques can be tested by projecting the original variables into a new variable space or using techniques available in the literature for the data used in neural networks.

- The IIoT Anomaly Classification Framework could be improved by adding a task directly related to techniques for models to be automatically retrained and improved while in production.

- New framework implementations should include a greater diversity of failures and attacks that could affect IIoT. It would be a good idea to create a taxonomy of faults and attacks for IIoT with recommendations on how to simulate and publish them online.

This thesis presents a robust anomaly detection and classification framework in Industrial Internet of Things (IIoT). The framework's adaptability and continuous improvement capabilities, validated through cross-validation, underscore its potential for real-world applications. The next chapter will present the conclusions and outline future research directions.

Chapter 7

Conclusion and Future Work

The rapidly evolving landscape of data-intensive environments needs robust anomaly detection and classification frameworks to ensure system reliability and security. This thesis proposes a dual-model approach, which employs separate anomaly detection and classification models and represents a strategic advancement in addressing these challenges. This framework leverages the strengths of specialized models to enhance the accuracy and efficiency of anomaly identification and categorization. This approach offers a comprehensive solution that is adaptable to various applications by focusing on detecting and classifying anomalies. The following conclusion summarizes the contributions and research work in the thesis and some future research lines.

7.1 Conclusion

This thesis has successfully demonstrated the efficacy of a dual-model framework for anomaly detection and classification in data-intensive environments, such as those encountered in IIoT systems. The separate but complementary models for detecting and classifying anomalies leverage advanced machine learning techniques, optimized processing methods, and contextual integration to significantly enhance anomaly management's accuracy and efficiency. This framework shows promise in adapting to the dynamic nature of IIoT environments, ensuring its long-term applicability and effectiveness. Throughout this work, we have demonstrated the importance of several key components:

- According to the latest research, anomaly classification in Industrial Internet of Things (IIoT) is still an ongoing area of study. Out of a hundred articles reviewed, only 13% provide some form of categorization, focusing mainly on industrial process differentiation and types of failures or attacks. In comparison, 51% of the articles focus on attack detection, while 38% address anomalies in general.

- The comprehensive framework describes a new approach to detecting and categorizing anomalies in IIoT environments. Combines CPS domain knowledge with data-driven insights from IIoT technologies. The choice of machine learning algorithms, tailored to the specific characteristics of the dataset and the nature of anomalies, plays a significant role in the framework's success. To identify the most suitable techniques for different anomaly detection scenarios, we explored models, including supervised, unsupervised, and semi-supervised learning approaches.

- Anomaly detector and classifier two-stage model: The first model is designed to detect anomalies. This model can detect new anomalies by only identifying deviations from the norm. Once anomalies are detected, a separate classification model is employed to categorize these anomalies into failure and attack classes. This model leverages supervised learning techniques, utilizing labeled data to learn and predict the nature of the anomalies. Separating detection and classification tasks allows for more specialized and accurate classification results. The cross-validation obtained a precision equal to 0.88, a recall of 0.93, and an F1-score of 0.90. Furthermore, false positives and negatives occurred around the anomalies used to evaluate the system, which did not generate false alarms that represented additional effort for the system's human operators.

- The evaluation of the anomaly classifier with an anomaly (sensor disconnection failure) not used to train the models showed that the detector identifies it. Still, the classifier labels it as an 'event' so that the human operator can proceed to manage it. This is considered a success of the framework because it can detect new anomalies.

- Nearly half (48.9%) of the reviewed papers in the state-of-the-art focus on deploying anomaly detection systems on edge devices. This reflects the industry's shift towards edge computing, aiming to reduce latency and enhance real-time decision-making. The experiment evaluating the light versions of the detection and classification models running on a Raspberry PI3 showed similar metrics to the original models running on Google Colaboratory: 0.91 for precision, 0.90 for recall, and 0.90 for F1-score.

- Context Information Integration: Incorporating context information significantly enhances the framework's ability to classify anomalies accurately. Contextual data, such as temporal patterns, environmental conditions, and domain-specific factors, provide additional layers of information that improve the model's decision-making process. By integrating this contextual information, we can better distinguish between true anomalies and benign deviations, thereby reducing false positives and improving overall detection accuracy. In chapter 5, deep learning models were trained with and without context information. The results showed that in the anomaly detector model, F1-score went from 0.92 to 0.96 and from 0.83 to 0.90 in the classifier when they used the speeds of the industrial process motors as context information.

- Evaluation Metrics: Implementing appropriate evaluation metrics is essential for assessing model performance and ensuring the reliability of the classification results. We emphasized using precision, recall, and F1-score metrics to evaluate the framework comprehensively because none of these metrics use the True Negative rate (normal data) in their equations. Since the number of normal samples far exceeds the number of anomalies, the metrics, including TN, favor the normal class.

- Scalability and Real-Time Processing: The ability to scale the framework and process data in real-time is critical for practical applications. We discussed strategies for optimizing the framework to handle large volumes of data and deliver timely anomaly detection results, ensuring its applicability in dynamic environments.

- Continuous Improvement: Anomaly classification is an ongoing process that requires continuous monitoring and model updates. We highlighted the importance of incorporating feedback loops, periodic retraining, and adapting to evolving data patterns to maintain the framework's effectiveness over time.

The proposed framework addresses the challenges associated with anomaly classification and provides a comprehensive solution that can be adapted to various applications, from cybersecurity to industrial monitoring. By integrating advanced machine learning techniques with robust preprocessing and evaluation methods, this framework offers a powerful tool for identifying and responding to anomalies.

7.2 Future Work

Various interesting paths can be explored to enhance the framework's performance. One such way is to focus on overcoming the current limitations of the methods, which can include factors such as scalability, adaptability, and robustness. Additionally, the developed framework has the potential for further extensions, such as incorporating new data sources and exploring alternative algorithmic approaches. Some lines of work summarized below offer exciting opportunities for researchers and practitioners to continue advancing the field and improving the effectiveness of the methods.

- Heterogeneity of Data Sources. Training detection and classification models with diverse data sources such as time series, images, thermography, videos, sounds, and production data can improve their accuracy and decision-making capabilities. Exposing models to various data inputs helps them better recognize failures and attacks.

- Distributed Architecture. The anomaly classification framework addresses anomalies in Industrial Internet of Things (IIoT) systems. Implementing the framework in a distributed architecture enhances adaptability, scalability, and the ability to recognize local and global anomalies. Organizations could train ML models in a distributed manner using federated learning techniques to recognize and recover from individual node anomalies. Implementing this framework requires careful planning, attention to detail, and a deep understanding of IIoT, machine learning, and distributed systems.

- Incorporate Specific Domain Knowledge. Incorporating specific domain knowledge into anomaly classification in an IIoT system enhances accuracy. Consider the operational context and environmental factors, and use expert rules and machine learning. Utilize physical laws governing CPS operations for richer data sets.

- Retraining Models. Updating models regularly with new data and insights from ongoing operations and expert feedback is important. This helps the system adapt to changes in its environment. Experts can provide valuable insights into anomalies and their severity, so it is necessary to establish a mechanism to provide feedback on the system's predictions. Incorporating historical data about past failures, attacks, and successful operations can refine model predictions and improve system accuracy.

- Test advanced techniques for variable selection, data balancing, and hyperparameter optimization for enhancing the robustness and accuracy of predictive models. Specifically, exploring sophisticated techniques for variable selection can aid in identifying the most influential predictors, thereby simplifying models and improving interpretability. Addressing data imbalances through advanced balancing techniques such as synthetic data generation or adaptive sampling could significantly improve model performance, especially in skewed class distributions. Furthermore, refining hyperparameter optimization strategies, potentially through automated machine learning frameworks, would streamline the model development process and potentially reveal more optimal configurations that manual tuning might miss. These advanced techniques collectively promise to elevate the efficiency and efficacy of future modeling efforts, offering a more nuanced understanding and application in predictive analytics.

- Improving Computational Efficiency. In the Industrial Internet of Things (IIoT) realm, edge devices are crucial in managing and processing large amounts of data generated by industrial processes. These devices are positioned at the boundary between the industrial environment and the broader network infrastructure and are responsible for real-time data analysis to identify anomalies. To optimize anomaly classification on edge devices, algorithm complexity can be refined, less resource-intensive models can be implemented, and hardware acceleration using specialized hardware such as FPGAs or GPUs can be utilized. These optimizations ensure that the processing capabilities of edge devices are maximally utilized without compromising speed and accuracy.

- Customized Alerts and Responses. Design an alert system to provide actionable, contextual information and prioritize it based on anomaly severity and type. Create automated or manual intervention protocols based on the anomaly type to minimize downtime and prevent damage. This framework should be an integral part of the operational process, offering insights and actions tailored to the specific needs and challenges of the monitored CPS.

- Expansion to Other Domains. The IIoT anomaly classification framework can benefit from customization to address different industrial sectors' unique needs and challenges. Customizing the anomaly detection algorithms to consider specific types of equipment, failure modes, and environmental factors pertinent to each sector can enhance the accuracy and relevance of anomaly detection. Sector-specific pilot studies could be implemented to validate and refine the adaptability of the IIoT anomaly classification framework across various domains. Comparative studies across different sectors could provide insights into the framework's versatility and areas for improvement.

- Self-Improving Systems. The goal is to create a seamless and automated workflow within IIoT environments that detects anomalies and takes intelligent actions to mitigate risks efficiently. This proactive approach to industrial operations can significantly enhance reliability, reduce costs, and improve safety standards. Feedback mechanisms classify anomalies, improving adaptability and efficiency over time. Machine learning models adjust their behavior based on real-world feedback, refining detection algorithms and enhancing system reliability. Establishing a robust loop where the system reacts to anomalies and records the outcomes of its actions is essential for effective feedback mechanisms.
- More Data and Diverse Anomalies. Testing the IIoT anomaly classification framework with larger and more varied datasets is crucial for enhancing robustness and minimizing overfitting. Incorporating diverse scenarios and data types into the training process helps the model generalize better across different conditions and equipment types in IIoT systems, improving accuracy and reliability. Extensive testing with comprehensive datasets is fundamental for developing a resilient anomaly detection system that can deliver consistent and reliable results across a broad spectrum of IIoT environments. Incorporating a broad spectrum of anomalies into the testing phase helps the framework recognize and react to various situations, improving reliability.
- Theoretical Advancements. Theoretical studies are crucial in understanding anomaly classification within the Industrial Internet of Things. Researchers can design more effective anomaly detection systems by delving into underlying principles and mathematical models. This academic pursuit enhances theoretical understanding and fosters innovation for practical applications. Encouraging more theoretical research could transform the evaluation and development of anomaly detection models, leading to industry standards and improved systems.
- Security Enhancements. Ensuring the security of the IIoT anomaly classification framework is crucial to protect IIoT environments from adversarial attacks. As these systems become increasingly essential for critical infrastructure operations,

they become tempting targets for malicious entities seeking to disrupt operational processes or gain unauthorized access to sensitive data. Adversarial attacks, such as data poisoning or model evasion, can manipulate anomaly detection, resulting in incorrect classifications, missed detections, and wrong automated responses. Strengthening the security of the anomaly classification framework requires deploying robust mechanisms to detect and counter such attacks, ensuring the system's outputs' integrity and reliability. This involves implementing techniques like anomaly detection specific to adversarial attempts, which can identify unusual patterns in the input data that may indicate tampering or intrusion.

- Collaborative and Interdisciplinary Research. Partnering with industrial entities lets them gain real-world insights and validate the IIoT anomaly classification framework. These collaborations provide access to diverse industrial environments for testing the framework on actual data streams. They also allow for incorporating domain-specific knowledge and operator feedback, ensuring effective and robust anomaly detection and response mechanisms tailored to industry needs.

The potential for further research and development in this field is vast. By pursuing these avenues, researchers can continue to advance the field of anomaly detection, making robust, adaptable systems that can meet the evolving challenges of modern industrial environments.

Appendix A

Appendix for Chapter 5

A.1 Sequence Creation Code

The time series is transformed from the form [n_samples, n_features] to [n_samples, n_times, n_features] before feeding the detection and classification models; this allows to detect the correlation between the samples. The following function allows you to perform this transformation.

```python
def create_sequences(values, n_samples, n_times, n_features):
  output = np.zeros( (n_samples-n_times, n_times, n_features) )
  for j in range(np.array(n_features)):
    features = []
    for i in range(np.array(n_samples - n_times)):
      output[i,:,j]=values[i : (i + n_times),j]
  return output
```

Listing A.1: Generating training sequence

The following is a snippet of code used to invoke the function that generates the training sequence.

```python
df_training = np.array(x_train[:])
N_SAMPLES= df_training.shape[0]
N_FEATURES = df_training.shape[1]
x_train_sequence= create_sequences(np.array(df_training), np.array(
    N_SAMPLES), np.array(N_TIMES), np.array(N_FEATURES))
x_train_sequence = x_train_sequence.astype(np.float32)
```

Listing A.2: Invoking training sequence

A.2 Predicting and Classifying Anomalies Code

The code shown below invokes the anomaly detector model. If the number of samples labeled as anomalies exceeds 15, the anomaly classifier is invoked.

```
# Predicting and Classifing Anomalies
anomaly = ['A', 'F', 'E']
N_TIMES = 300
pred_detect = []
classes = []
with tf.device('/device:GPU:0'):
  i = N_TIMES
  while i < (x_test.shape[0] - 1):
    df_original = np.expand_dims(x_test[i-N_TIMES:i], axis=0)
    diff = np.abs(detector.predict([df_original], verbose=False) - 
   df_original)
    diff = diff.reshape(diff.shape[1], diff.shape[2])
    pred_detect += list(np.sum(diff > threshold, axis=1)/(np.sum(diff > 
    threshold, axis=1)+0.0001))

    if np.sum(pred_detect[i-N_TIMES:i]) > 15:
      predicted = classifier.predict([df_original], verbose=False)
      predicted = predicted.reshape(predicted.shape[1], predicted.shape
   [2])
      classes += list([anomaly[np.argmax(predicted, axis=1)[j]] for j 
   in range(N_TIMES)])
    else:
      classes += list(np.full(N_TIMES, 'N'))
    i+=N_TIMES
# Show metrics
true_labels = list(df.iloc[:len(classes),22])
predicted_labels = classes
print(classification_report(true_labels, predicted_labels))
```

Listing A.3: Predicting and Classifying Anomalies Code

Appendix B

Appendix for Chapter 6

B.1 IIoT Anomaly Detector Code

The following is the code that outlines the architecture of the autoencoder used as an anomaly detector in the framework's implementation after optimizing the number of neurons in the external and internal layers.

```
# Autoencoder CONV1D Model
def autoencoder_model(x_train):
  import tensorflow as tf
  from numpy import array
  from keras.models import Sequential
  from keras.layers import Dense
  from keras.layers import Conv1D
  model = keras.Sequential(
    [
      layers.Conv1D( filters=64, kernel_size=5, padding="same", strides
    =1, activation="relu", input_shape=(x_train.shape[1], x_train.shape
    [2])),
      layers.Dropout(rate=0.2),
      layers.Conv1D( filters=32, kernel_size=5, padding="same", strides
    =1, activation="relu" ),
      layers.Dropout(rate=0.2),
      layers.Conv1D( filters=7, kernel_size=5, padding="same", strides
    =1, activation="relu" ),

      layers.Conv1DTranspose( filters=7, kernel_size=5, padding="same",
     strides=1, activation="relu" ),
      layers.Dropout(rate=0.2),
      layers.Conv1DTranspose( filters=32, kernel_size=5, padding="same"
    , strides=1, activation="relu" ),
      layers.Dropout(rate=0.2),
      layers.Conv1DTranspose( filters=64, kernel_size=5, padding="same"
    , strides=1, activation="relu" ),
      layers.Dense(units=x_train.shape[2], activation='sigmoid')
    ]
  )
  return model
```

Listing B.1: Detector Model. Autoencoder code

B.2 IIoT Anomaly Classifier Code

The following code outlines the architecture of the transformer used as an anomaly classifier in the framework's implementation. This model uses three encoders and 20 multi-head attentions.

```
  # Construct the transformer model
total_heads=20
sequence_length = x_train_sequence.shape[1] #timestep
embed_dim = x_train_sequence.shape[2]       #variables in dataset
classes = y_train_sequence.shape[2]  # A-F-N
dense_units = embed_dim
# Custom layers
embedding_layer = EmbeddingLayer(sequence_length, embed_dim)
encoder_layer = EncoderLayer(total_heads, dense_units, sequence_length,
    embed_dim, classes)
classify_layer = Dense(classes, activation='softmax')

# Connecting the layers together
inputs = Input(shape=(sequence_length,embed_dim ))
encoder_1 = encoder_layer(inputs)
encoder_2 = encoder_layer(encoder_1)
encoder_3 = encoder_layer(encoder_2)

dense = Dense(dense_units, activation="relu")(encoder_3)
outputs = classify_layer(dense)

model = Model(inputs=inputs, outputs=outputs)
model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=1e-4),
   loss='categorical_crossentropy')
```

```
model.save("transformer.keras")
model.summary()
```

Listing B.2: Detector Model. Autoencoder code

B.3 IIoT Anomaly Classification Model Validation

Figure B.1 shows the graphical interpretation of the classification results during cross-validation for the edge1 device. All anomalies were detected; some anomalies present few samples labeled as events, and in the edge1_13 set, the number of samples erroneously labeled as events increases slightly. Thanks to the validation of 15 samples (3 seconds) in the input sequence, the classifier (in the middle) eliminates the false positives not associated with real anomalies generated in the detector (on the left). The plots on the right show the actual anomalies.

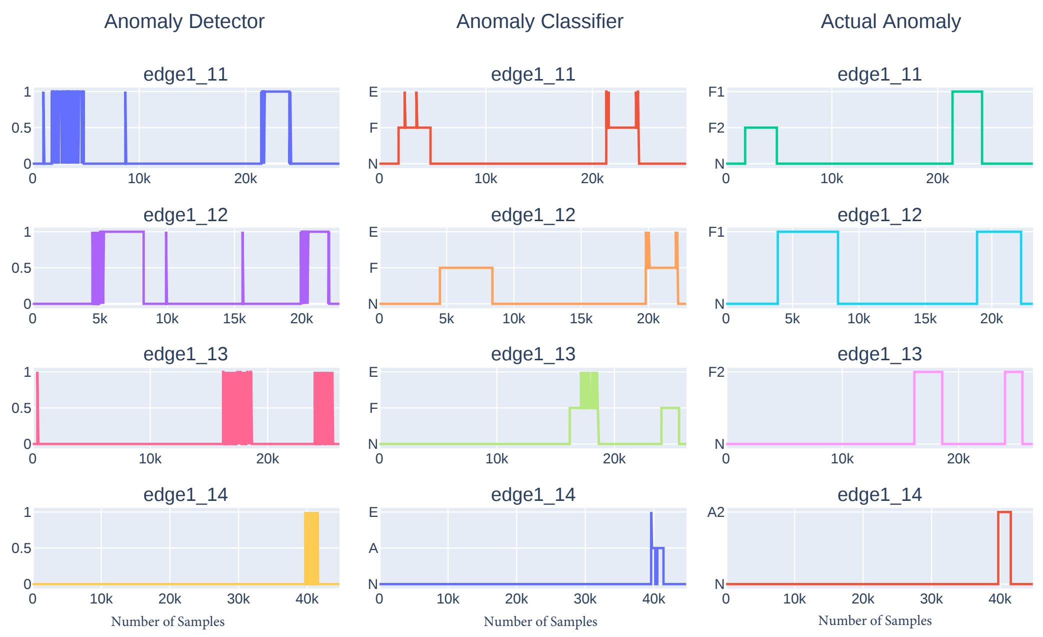

Figure B.1: Graphical result for cross-validation with edge1 device data.

Figure B.2 shows edge2 device anomalies successfully detected with a few samples classified as events. Remember that *event* occurs when the classifier considers that a sample labeled by the detector as an anomaly is normal.

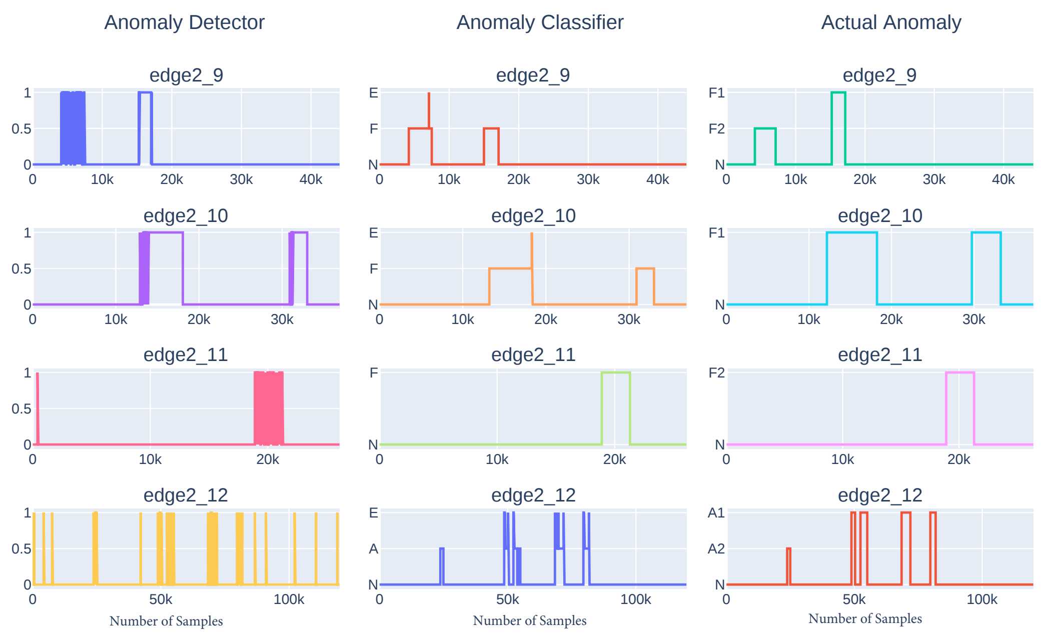

Figure B.2: Graphical result for cross-validation with edge2 device data.

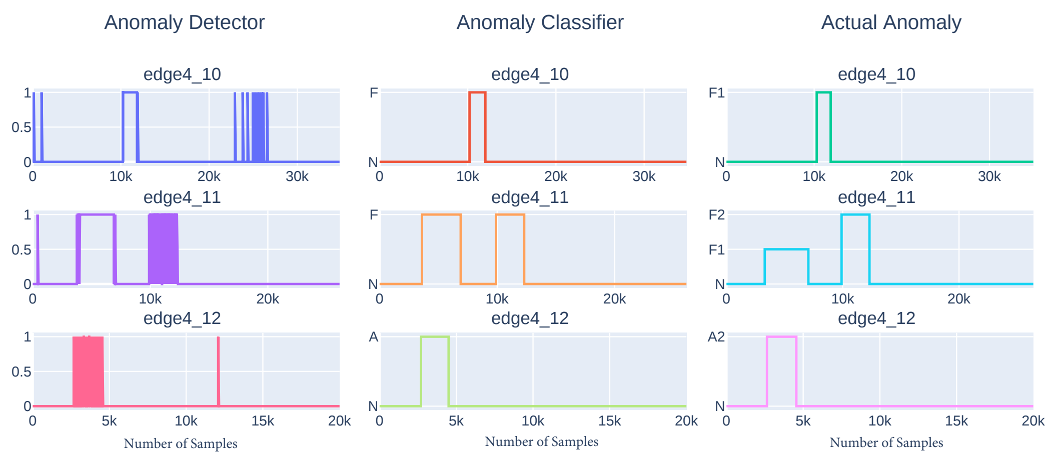

Figure B.3: Graphical result for cross-validation with edge4 device data.

Figure B.4 shows a similar result for the other edge devices.

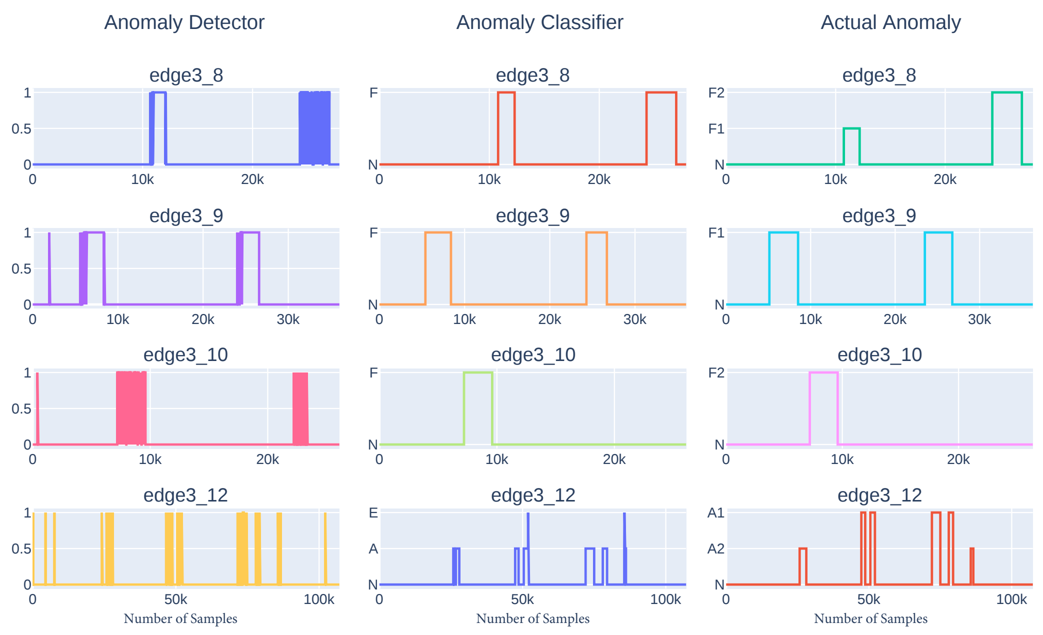

Figure B.4: Graphical result for cross-validation with edge3 device data.

Figure B.3 shows that all anomalies have been detected and classified correctly. Even a false positive detected by the detector was omitted by the classifier by applying the rule of 15 positive samples in the sequence of 300 samples.

B.4 Deploying the IIoT Anomaly Classifier in the Testbed

The code shown in B.3 corresponds to a function in Python that receives the sequence of 300 samples and passes it to the detector. If it predicts 15 samples or more as anomalies, the detector invokes the classifier model to label them as failures or attacks. But if the classifier considers one of these samples normal, the function labels it as an event so that a human operator can classify it manually.

```python
def predicting(sequence):
  anomaly = ['A', 'F', 'E']
  interpreter_d.set_tensor(input_details_d[0]['index'], sequence)
  interpreter_d.invoke()
  prediction_d = interpreter_d.get_tensor(output_details_d[0]['index'])[0]
  diff = np.abs(prediction_d - sequence.reshape(sequence.shape[1], sequence.shape[2]))
  pred_detect = list(np.sum(diff > threshold, axis=1)/(np.sum(diff > threshold, axis=1)+0.0001))
  if np.sum(pred_detect[i-N_TIMES:i]) > 15:
    interpreter_c.set_tensor(input_details_c[0]['index'], sequence)
    interpreter_c.invoke()
    predicted = interpreter_c.get_tensor(output_details_c[0]['index'])[0]
    return pred_detect, list([anomaly[np.argmax(predicted, axis=1)[j]] for j in range(N_TIMES)])
  else:
    return pred_detect, list(np.full(N_TIMES, 'N'))
```

Listing B.3: Function to invoke detector and classifier models on Raspberry PI3.

The previous function can be invoked from code similar to the following, where x_test is a dataframe with a test dataset.

```
from sklearn.metrics import classification_report
N_TIMES = 300
pred_detect = []
classes = []
i = N_TIMES
while i < (x_test.shape[0] - 1):
  df_original = np.expand_dims(x_test[i-N_TIMES:i].astype(np.float32), axis=0)
  classify = predicting(df_original)
  pred_detect += classify[0]
  classes += classify[1]
  i+=N_TIMES
print(len(classes))
true_labels = list(df.iloc[:len(classes),22])
predicted_labels = classes
print(classification_report(true_labels, predicted_labels))
```

Listing B.4: Code to invoke predicting() function.

References

[1] H. Benaddi, M. Jouhari, K. Ibrahimi, J. Ben Othman, and E. M. Amhoud, “Anomaly detection in industrial iot using distributional reinforcement learning and generative adversarial networks,” *Sensors*, vol. 22, no. 21, p. 8085, 2022. (pages 1, 10, and 42)

[2] J. Li, M. S. Othman, H. Chen, and L. M. Yusuf, “Optimizing iot intrusion detection system: feature selection versus feature extraction in machine learning,” *Journal of Big Data*, vol. 11, no. 1, p. 36, 2024. (pages 16, 20, 21, 23, 25, 26, 34, 44, 45, 66, 68, 69, 70, and 109)

[3] X. Liu and Y. Du, “Towards effective feature selection for iot botnet attack detection using a genetic algorithm,” *Electronics*, vol. 12, no. 5, p. 1260, 2023. (pages 1, 10, 34, and 42)

[4] A. Karkouch, H. Mousannif, H. Al Moatassime, and T. Noel, “Data quality in internet of things: A state-of-the-art survey,” *Journal of Network and Computer Applications*, vol. 73, pp. 57–81, 2016. (pages 1, 10, 11, 13, 22, 23, 24, 65, 67, and 68)

[5] J. Byabazaire, G. O’Hare, and D. Delaney, “Data quality and trust: Review of challenges and opportunities for data sharing in iot,” *Electronics*, vol. 9, no. 12, p. 2083, 2020. (pages 11, 22, and 65)

[6] M. Soori, B. Arezoo, and R. Dastres, “Internet of things for smart factories in industry 4.0, a review,” *Internet of Things and Cyber-Physical Systems*, 2023. (pages 1, 2, 10, 11, 56, and 60)

[7] T. Qiu, J. Chi, X. Zhou, Z. Ning, M. Atiquzzaman, and D. O. Wu, “Edge computing in industrial internet of things: Architecture, advances and challenges,” *IEEE Communications Surveys & Tutorials*, vol. 22, no. 4, pp. 2462–2488, 2020. (pages 11 and 12)

[8] A. A. Mirani, G. Velasco-Hernandez, A. Awasthi, and J. Walsh, “Key challenges and emerging technologies in industrial iot architectures: A review,” *Sensors*, vol. 22, no. 15, p. 5836, 2022. (pages 11 and 56)

[9] H. Nizam, S. Zafar, Z. Lv, F. Wang, and X. Hu, “Real-time deep anomaly detection framework for multivariate time-series data in industrial iot,” *IEEE Sensors Journal*, vol. 22, no. 23, pp. 22836–22849, 2022. (pages 1, 2, 10, 11, 12, 15, 16, 18, 19, 26, 34, 35, 37, 46, 52, 60, 69, 72, and 109)

[10] H. Zhang, Y. Xia, T. Yan, and G. Liu, “Unsupervised anomaly detection in multivariate time series through transformer-based variational autoencoder,” in *2021 33rd Chinese Control and Decision Conference (CCDC)*, pp. 281–286, IEEE, 2021. (pages 1, 7, 9, 10, 14, 18, 20, 41, 42, 47, 56, and 59)

[11] S. Tuli, G. Casale, and N. R. Jennings, “Tranad: Deep transformer networks for anomaly detection in multivariate time series data,” *arXiv preprint arXiv:2201.07284*, 2022. (pages 1, 13, 14, 18, 26, 27, 34, 42, 48, 68, 69, and 109)

[12] H. T. Truong, B. P. Ta, Q. A. Le, D. M. Nguyen, C. T. Le, H. X. Nguyen, H. T. Do, H. T. Nguyen, and K. P. Tran, “Light-weight federated learning-based anomaly detection for time-series data in industrial control systems,” *Computers in Industry*, vol. 140, p. 103692, 2022.
(pages 1, 2, 9, 10, 11, 14, 34, 37, 41, 45, 46, 47, 52, 54, 56, 59, 60, 68, 69, and 70)

[13] C. Ding, J. Zhao, and S. Sun, “Concept drift adaptation for time series anomaly detection via transformer,” *Neural Processing Letters*, vol. 55, no. 3, pp. 2081–2101, 2023. (pages 1, 14, 18, 34, 41, and 47)

[14] M. Rodríguez, D. P. Tobón, and D. Múnera, “Anomaly classification in industrial internet of things: A review,” *Intelligent Systems with Applications*, p. 200232, 2023. (pages 2, 6, 37, 42, and 51)

[15] B. Yang, J. Ye, S. Coshatt, W. Song, and F. Zahiri, “Data-driven approach for detection of physical faults and cyber attacks in manufacturing motor drives,” in *2022 IEEE Energy Conversion Congress and Exposition (ECCE)*, pp. 1–6, IEEE, 2022. (pages 2 and 49)

[16] K. Zhang, C. Keliris, T. Parisini, and M. M. Polycarpou, “Identification of sensor replay attacks and physical faults for cyber-physical systems,” *IEEE Control Systems Letters*, vol. 6, pp. 1178–1183, 2021. (pages 2, 15, and 49)

[17] M. Ganjkhani, M. Gilanifar, J. Giraldo, and M. Parvania, “Integrated cyber and physical anomaly location and classification in power distribution systems,” *IEEE Transactions on Industrial Informatics*, vol. 17, no. 10, pp. 7040–7049, 2021. (pages 2 and 49)

[18] L. Liang and S. Liu, “Event-triggered distributed attack detection and fault diagnosis,” *IEEE Transactions on Instrumentation and Measurement*, vol. 72, pp. 1–11, 2022. (pages 2 and 50)

[19] C. EUREKA, G. de Antioquia, *et al.*, “Engineering for transformation,” in *Expo Ingenieria*, Fondo Editorial EIA, 2022. (pages 6 and 107)

[20] M. Ryalat, H. ElMoaqet, and M. AlFaouri, “Design of a smart factory based on cyber-physical systems and internet of things towards industry 4.0,” *Applied Sciences*, vol. 13, no. 4, p. 2156, 2023. (pages 7, 9, and 56)

[21] Y. Gao, X. Wang, X. He, Z. Liu, H. Feng, and Y. Zhang, “Addressing heterophily in graph anomaly detection: A perspective of graph spectrum,” in *Proceedings of the ACM Web Conference 2023*, pp. 1528–1538, 2023. (pages 7 and 12)

[22] D. J. Atul, R. Kamalraj, G. Ramesh, K. S. Sankaran, S. Sharma, and S. Khasim, "A machine learning based iot for providing an intrusion detection system for security.," *Microprocess. Microsystems*, vol. 82, p. 103741, 2021. (pages 7, 12, 51, 59, 72, and 109)

[23] Z. Niu, W. Guo, J. Xue, Y. Wang, Z. Kong, and L. Huang, "A novel anomaly detection approach based on ensemble semi-supervised active learning (adessa)," *Computers & Security*, vol. 129, p. 103190, 2023. (pages 9, 10, 17, 18, 44, 45, 56, 59, and 109)

[24] A. Ba, K. Lynch, J. Ploennigs, B. Schaper, C. Lohse, and F. Lorenzi, "Automated configuration of heterogeneous graph neural networks with a semantic math parser for iot systems," *IEEE Internet of Things Journal*, vol. 10, no. 2, pp. 1042–1052, 2022. (pages 10, 45, 46, 47, 52, and 59)

[25] A. A. Hanna and W. Jasweijer, "Modeling domain knowledge using explicit conceptualization," *IEEE Expert: Intelligent Systems and Their Applications*, pp. 53–64, 1994. (pages 10 and 59)

[26] A. Moradbeikie, K. Jamshidi, A. Bohlooli, J. Garcia, and X. Masip-Bruin, "An iiot based ics to improve safety through fast and accurate hazard detection and differentiation," *IEEE access*, vol. 8, pp. 206942–206957, 2020. (pages 10, 49, and 51)

[27] M. Younan, E. H. Houssein, M. Elhoseny, and A. A. Ali, "Challenges and recommended technologies for the industrial internet of things: A comprehensive review," *Measurement*, vol. 151, p. 107198, 2020. (pages 10, 11, 37, 56, and 60)

[28] J. Lin, W. Yu, N. Zhang, X. Yang, H. Zhang, and W. Zhao, "A survey on internet of things: Architecture, enabling technologies, security and privacy, and applications," *IEEE internet of things journal*, vol. 4, no. 5, pp. 1125–1142, 2017. (pages 11 and 12)

[29] Z. Li, Y. Zhu, and M. Van Leeuwen, "A survey on explainable anomaly detection," *ACM Transactions on Knowledge Discovery from Data*, vol. 18, no. 1, pp. 1–54, 2023. (pages 13, 14, and 20)

[30] P. Yan, A. Abdulkadir, P.-P. Luley, M. Rosenthal, G. A. Schatte, B. F. Grewe, and T. Stadelmann, "A comprehensive survey of deep transfer learning for anomaly detection in industrial time series: Methods, applications, and directions," *IEEE Access*, 2024. (pages 13, 18, 20, 27, 29, and 71)

[31] N. Ghosh, K. Maity, R. Paul, and S. Maity, "Outlier detection in sensor data using machine learning techniques for iot framework and wireless sensor networks: A brief study," in *2019 International Conference on Applied Machine Learning (ICAML)*, pp. 187–190, IEEE, 2019. (pages 13 and 62)

[32] F. Cauteruccio, L. Cinelli, E. Corradini, G. Terracina, D. Ursino, L. Virgili, C. Savaglio, A. Liotta, and G. Fortino, "A framework for anomaly detection and classification in multiple iot scenarios," *Future Generation Computer Systems*, vol. 114, pp. 322–335, 2021. (pages 13, 15, 16, and 51)

[33] T. Çavdar, N. Ebrahimpour, M. T. Kakız, and F. B. Günay, "Decision-making for the anomalies in iiots based on 1d convolutional neural networks and dempster–shafer theory (ds-1dcnn)," *The Journal of Supercomputing*, vol. 79, no. 2, pp. 1683–1704, 2023. (pages 13, 14, 17, 19, 41, 42, 45, 46, 47, 48, 52, and 62)

[34] K. DeMedeiros, A. Hendawi, and M. Alvarez, "A survey of ai-based anomaly detection in iot and sensor networks," *Sensors*, vol. 23, no. 3, p. 1352, 2023. (pages 13, 14, 30, 32, 33, 34, and 73)

[35] M. A. Belay, S. S. Blakseth, A. Rasheed, and P. Salvo Rossi, "Unsupervised anomaly detection for iot-based multivariate time series: Existing solutions, performance analysis and future directions," *Sensors*, vol. 23, no. 5, p. 2844, 2023. (pages 13, 15, 16, 17, 19, 34, and 109)

[36] S. Saurav, P. Malhotra, V. TV, N. Gugulothu, L. Vig, P. Agarwal, and G. Shroff, "Online anomaly detection with concept drift adaptation using recurrent neural networks," in *Proceedings of the acm india joint international conference on data science and management of data*, pp. 78–87, 2018. (pages 13, 46, and 52)

[37] J. Kim, H. Kang, and P. Kang, "Time-series anomaly detection with stacked transformer representations and 1d convolutional network," *Engineering Applications of Artificial Intelligence*, vol. 120, p. 105964, 2023. (pages 14, 15, 17, 19, 26, 27, 34, 45, 46, 47, 52, 68, 69, and 109)

[38] A. N. Tarekegn, M. Giacobini, and K. Michalak, "A review of methods for imbalanced multi-label classification," *Pattern Recognition*, vol. 118, p. 107965, 2021. (pages 15, 27, 28, 30, 31, 32, 33, 34, 71, and 73)

[39] M. Panahandeh, A. Hamou-Lhadj, M. Hamdaqa, and J. Miller, "Serviceanomaly: An anomaly detection approach in microservices using distributed traces and profiling metrics," *Journal of Systems and Software*, vol. 209, p. 111917, 2024. (pages 15, 31, 32, 33, 34, and 73)

[40] S. Handoyo, Y.-P. Chen, G. Irianto, and A. Widodo, "The varying threshold values of logistic regression and linear discriminant for classifying fraudulent firm," *Mathematics and Statistics*, vol. 9, no. 2, pp. 135–143, 2021. (pages 15, 16, 19, and 34)

[41] J. Kaur, K. S. Parmar, and S. Singh, "Autoregressive models in environmental forecasting time series: a theoretical and application review," *Environmental Science and Pollution Research*, vol. 30, no. 8, pp. 19617–19641, 2023. (page 15)

[42] I. Silva and J. Eugenio Naranjo, "A systematic methodology to evaluate prediction models for driving style classification," *Sensors*, vol. 20, no. 6, p. 1692, 2020. (pages 15, 27, 34, and 71)

[43] A. Yeganeh, N. Chukhrova, A. Johannssen, and H. Fotuhi, "A network surveillance approach using machine learning based control charts," *Expert Systems with Applications*, vol. 219, p. 119660, 2023. (pages 16 and 43)

[44] C.-I. Chang, C.-Y. Lin, P.-C. Chung, and P. F. Hu, "Iterative spectral–spatial hyperspectral anomaly detection," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 61, pp. 1–30, 2023. (pages 16 and 43)

[45] F. A. Mazarbhuiya and M. Shenify, “A mixed clustering approach for real-time anomaly detection,” *Applied Sciences*, vol. 13, no. 7, p. 4151, 2023. (pages 16, 43, and 109)

[46] J. Jithish, B. Alangot, N. Mahalingam, and K. S. Yeo, “Distributed anomaly detection in smart grids: a federated learning-based approach,” *IEEE Access*, vol. 11, pp. 7157–7179, 2023. (pages 18, 23, 34, 48, 66, and 109)

[47] X. Sáez-de Cámara, J. L. Flores, C. Arellano, A. Urbieta, and U. Zurutuza, “Clustered federated learning architecture for network anomaly detection in large scale heterogeneous iot networks,” *Computers & Security*, vol. 131, p. 103299, 2023. (pages 16, 18, 34, 48, and 109)

[48] Y. Li, X. Peng, J. Zhang, Z. Li, and M. Wen, “Dct-gan: dilated convolutional transformer-based gan for time series anomaly detection,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 4, pp. 3632–3644, 2021. (pages 17, 18, 26, 27, 34, 48, and 69)

[49] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, “Attention is all you need,” *Advances in neural information processing systems*, vol. 30, 2017. (pages 18, 47, 102, 103, and 112)

[50] F. Zeng, M. Chen, C. Qian, Y. Wang, Y. Zhou, and W. Tang, “Multivariate time series anomaly detection with adversarial transformer architecture in the internet of things,” *Future Generation Computer Systems*, vol. 144, pp. 244–255, 2023. (pages 18, 34, 48, 68, and 109)

[51] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial networks,” *Communications of the ACM*, vol. 63, no. 11, pp. 139–144, 2020. (page 18)

[52] X. Wang, Y. Wang, Z. Javaheri, L. Almutairi, N. Moghadamnejad, and O. S. Younes, “Federated deep learning for anomaly detection in the internet of things,” *Computers and Electrical Engineering*, vol. 108, p. 108651, 2023. (pages 18, 34, and 109)

[53] A. Chevrot, A. Vernotte, and B. Legeard, “Cae: Contextual auto-encoder for multivariate time-series anomaly detection in air transportation,” *Computers & Security*, vol. 116, p. 102652, 2022. (pages 18, 19, 48, 72, 98, and 109)

[54] I. Ko, D. Chambers, and E. Barrett, “Adaptable feature-selecting and threshold-moving complete autoencoder for ddos flood attack mitigation,” *Journal of Information Security and Applications*, vol. 55, p. 102647, 2020. (pages 19, 34, and 72)

[55] J. U. Ko, K. Na, J.-S. Oh, J. Kim, and B. D. Youn, “A new auto-encoder-based dynamic threshold to reduce false alarm rate for anomaly detection of steam turbines,” *Expert Systems with Applications*, vol. 189, p. 116094, 2022. (pages 19, 48, 49, and 109)

[56] G. Sun, J. Li, J. Dai, Z. Song, and F. Lang, “Feature selection for iot based on maximal information coefficient,” *Future Generation Computer Systems*, vol. 89, pp. 606–616, 2018. (pages 20, 21, and 70)

[57] P. Mooijman, C. Catal, B. Tekinerdogan, A. Lommen, and M. Blokland, "The effects of data balancing approaches: A case study," *Applied Soft Computing*, vol. 132, p. 109853, 2023. (pages 20, 27, 28, 67, and 71)

[58] M. Krawczak and G. Szkatuła, "An approach to dimensionality reduction in time series," *Information Sciences*, vol. 260, pp. 15–36, 2014. (page 21)

[59] M. Shafiq, Z. Tian, A. K. Bashir, X. Du, and M. Guizani, "Corrauc: a malicious bot-iot traffic detection method in iot network using machine-learning techniques," *IEEE Internet of Things Journal*, vol. 8, no. 5, pp. 3242–3254, 2020. (pages 21 and 109)

[60] Y. Fang, Y. Yao, X. Lin, J. Wang, and H. Zhai, "A feature selection based on genetic algorithm for intrusion detection of industrial control systems," *Computers & Security*, vol. 139, p. 103675, 2024. (pages 21, 25, 34, and 109)

[61] E. de Matos, R. T. Tiburski, C. R. Moratelli, S. Johann Filho, L. A. Amaral, G. Ramachandran, B. Krishnamachari, and F. Hessel, "Context information sharing for the internet of things: A survey," *Computer Networks*, vol. 166, p. 106988, 2020. (pages 22 and 65)

[62] C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos, "Context aware computing for the internet of things: A survey," *IEEE communications surveys & tutorials*, vol. 16, no. 1, pp. 414–454, 2013. (page 22)

[63] P. Pradeep and S. Krishnamoorthy, "The mom of context-aware systems: A survey," *Computer Communications*, vol. 137, pp. 44–69, 2019. (pages 22, 52, and 53)

[64] J. Wang, Y. Liu, P. Li, Z. Lin, S. Sindakis, and S. Aggarwal, "Overview of data quality: Examining the dimensions, antecedents, and impacts of data quality," *Journal of the Knowledge Economy*, pp. 1–20, 2023. (pages 23, 24, 67, and 68)

[65] A. D. Woods, D. Gerasimova, B. Van Dusen, J. Nissen, S. Bainter, A. Uzdavines, P. E. Davis-Kean, M. Halvorson, K. M. King, J. A. Logan, *et al.*, "Best practices for addressing missing data through multiple imputation," *Infant and Child Development*, vol. 33, no. 1, p. e2407, 2024. (pages 23 and 24)

[66] X. Zhang, X. Sun, L. Xia, S. Tao, and S. Xiang, "A matrix completion method for imputing missing values of process data," *Processes*, vol. 12, no. 4, p. 659, 2024. (page 24)

[67] S. Tiwaskar, M. Rashid, and P. Gokhale, "Impact of machine learning-based imputation techniques on medical datasets-a comparative analysis," *Multimedia Tools and Applications*, pp. 1–21, 2024. (page 24)

[68] G. Du, J. Zhang, N. Zhang, H. Wu, P. Wu, and S. Li, "Semi-supervised imbalanced multi-label classification with label propagation," *Pattern Recognition*, p. 110358, 2024. (pages 27 and 71)

[69] I. Araf, A. Idri, and I. Chairi, “Cost-sensitive learning for imbalanced medical data: a review,” *Artificial Intelligence Review*, vol. 57, no. 4, pp. 1–72, 2024. (pages 27, 28, and 71)

[70] L. Cui, Z. Dong, H. Xu, and D. Zhao, “Triplet attention-enhanced residual tree-inspired decision network: A hierarchical fault diagnosis model for unbalanced bearing datasets,” *Advanced Engineering Informatics*, vol. 59, p. 102322, 2024. (pages 27, 34, and 71)

[71] K. R. M. Fernando and C. P. Tsokos, “Dynamically weighted balanced loss: class imbalanced learning and confidence calibration of deep neural networks,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 7, pp. 2940–2951, 2021. (page 28)

[72] S. Subramanian, A. Rahimi, T. Baldwin, T. Cohn, and L. Frermann, “Fairness-aware class imbalanced learning,” *arXiv preprint arXiv:2109.10444*, 2021. (page 29)

[73] M. Hossin and M. N. Sulaiman, “A review on evaluation metrics for data classification evaluations,” *International journal of data mining & knowledge management process*, vol. 5, no. 2, p. 1, 2015. (pages 30, 31, 34, 35, and 73)

[74] M. Grandini, E. Bagli, and G. Visani, “Metrics for multi-class classification: an overview,” *arXiv preprint arXiv:2008.05756*, 2020. (pages 30, 32, 33, 34, and 73)

[75] J. Zhou, A. H. Gandomi, F. Chen, and A. Holzinger, “Evaluating the quality of machine learning explanations: A survey on methods and metrics,” *Electronics*, vol. 10, no. 5, p. 593, 2021. (pages 34 and 35)

[76] N. Mallik, E. Bergman, C. Hvarfner, D. Stoll, M. Janowski, M. Lindauer, L. Nardi, and F. Hutter, “Priorband: Practical hyperparameter optimization in the age of deep learning,” *Advances in Neural Information Processing Systems*, vol. 36, 2024. (pages 35 and 36)

[77] M. Chernigovskaya, A. Nahhas, A. Kharitonov, and K. Turowski, “Hyperparameter optimization in the context of smart manufacturing: a systematic literature review,” *Procedia Computer Science*, vol. 232, pp. 804–812, 2024. (pages 36 and 74)

[78] S. Yu, P.-L. Ma, B. Singh, S. Silva, and M. Pritchard, “Two-step hyperparameter optimization method: Accelerating hyperparameter search by using a fraction of a training dataset,” *Artificial Intelligence for the Earth Systems*, vol. 3, no. 1, p. e230013, 2024. (page 36)

[79] B. Si, Z. Ni, J. Xu, Y. Li, and F. Liu, “Interactive effects of hyperparameter optimization techniques and data characteristics on the performance of machine learning algorithms for building energy metamodeling,” *Case Studies in Thermal Engineering*, p. 104124, 2024. (pages 35, 36, 74, and 109)

[80] K. Petersen, S. Vakkalanka, and L. Kuzniarz, “Guidelines for conducting systematic mapping studies in software engineering: An update,” *Information and software technology*, vol. 64, pp. 1–18, 2015. (page 39)

[81] J. Cai, Q. Wang, J. Luo, Y. Liu, and L. Liao, "Capbad: Content-agnostic, payload-based anomaly detector for industrial control protocols," *IEEE Internet of Things Journal*, 2021. (pages 42, 43, and 52)

[82] T.-B. Dang, D.-T. Le, M. Kim, and H. Choo, "Neighboring information exploitation for anomaly detection in intelligent iot," in *International Conference on Future Data and Security Engineering*, pp. 260–271, Springer, 2021. (pages 42, 43, and 52)

[83] F. De Vita, D. Bruneo, and S. K. Das, "A semi-supervised bayesian anomaly detection technique for diagnosing faults in industrial iot systems," in *2021 IEEE International Conference on Smart Computing (SMARTCOMP)*, pp. 31–38, IEEE, 2021. (pages 42, 43, and 52)

[84] A. Gorbenko and V. Popov, "Abnormal behavioral pattern detection in closed-loop robotic systems for zero-day deceptive threats," in *2020 International Conference on Industrial Engineering, Applications and Manufacturing (ICIEAM)*, pp. 1–6, IEEE, 2020. (pages 42, 43, and 52)

[85] W. Aoudi and M. Almgren, "A scalable specification-agnostic multi-sensor anomaly detection system for iiot environments," *International Journal of Critical Infrastructure Protection*, vol. 30, pp. 1–8, 2020. (pages 42, 43, and 52)

[86] Z. Li, X. Ding, and H. Wang, "An effective constraint-based anomaly detection approach on multivariate time series," in *Asia-Pacific Web (APWeb) and Web-Age Information Management (WAIM) Joint International Conference on Web and Big Data*, pp. 61–69, Springer, 2020. (pages 42, 43, and 52)

[87] I. Garitano, M. Iturbe, E. Ezpeleta, and U. Zurutuza, "Who's there? evaluating data source integrity and veracity in iiot using multivariate statistical process control," in *Security and Privacy Trends in the Industrial Internet of Things*, pp. 181–198, Springer, 2019. (pages 42, 43, 52, and 53)

[88] M. A. Faisal, A. A. Cardenas, and A. Wool, "Profiling communications in industrial ip networks: Model complexity and anomaly detection," in *Security and Privacy Trends in the Industrial Internet of Things*, pp. 139–160, Springer, 2019. (pages 42, 43, and 52)

[89] B. Genge, P. Haller, and C. Enăchescu, "Anomaly detection in aging industrial internet of things," *IEEE Access*, vol. 7, pp. 74217–74230, 2019. (pages 42, 43, and 52)

[90] C. Wang, B. Wang, H. Liu, and H. Qu, "Anomaly detection for industrial control system based on autoencoder neural network," *Wireless Communications and Mobile Computing*, vol. 2020, 2020. (pages 42 and 45)

[91] G. Bernieri and F. Pascucci, "Improving security in industrial internet of things: a distributed intrusion detection methodology," in *Security and privacy trends in the industrial Internet of Things*, pp. 161–179, Springer, 2019. (pages 42, 43, and 52)

[92] G. Bernieri, M. Conti, and G. Pozzan, “Amon: an automaton monitor for industrial cyber-physical security,” in *Proceedings of the 14th International Conference on Availability, Reliability and Security*, pp. 1–10, 2019. (pages 42, 43, and 52)

[93] C. Enăchescu, H. Sándor, and B. Genge, “A multi-model-based approach to detect cyber stealth attacks in industrial internet of things,” in *2019 International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, pp. 1–6, IEEE, 2019. (pages 42, 43, and 52)

[94] H. R. Ghaeini, D. Antonioli, F. Brasser, A.-R. Sadeghi, and N. O. Tippenhauer, “State-aware anomaly detection for industrial control systems,” in *Proceedings of the 33rd Annual ACM Symposium on Applied Computing*, pp. 1620–1628, 2018. (pages 42, 43, and 52)

[95] E. Zugasti, M. Iturbe, I. Garitano, and U. Zurutuza, “Null is not always empty: Monitoring the null space for field-level anomaly detection in industrial iot environments,” in *2018 Global Internet of Things Summit (GIoTS)*, pp. 1–6, IEEE, 2018. (pages 42, 43, and 52)

[96] F. Hashmat, S. G. Abbas, S. Hina, G. A. Shah, T. Bakhshi, and W. Abbas, “An automated context-aware iot vulnerability assessment rule-set generator,” *Computer Communications*, vol. 186, pp. 133–152, 2022. (pages 43 and 52)

[97] G. Aruquipa and F. Diaz, “An iot architecture based on the control of bio inspired manufacturing system for the detection of anomalies with vibration sensors,” *Procedia Computer Science*, vol. 200, pp. 438–450, 2022. (pages 43 and 52)

[98] X. Wang, D. Pi, X. Zhang, H. Liu, and C. Guo, “Variational transformer-based anomaly detection approach for multivariate time series,” *Measurement*, p. 110791, 2022. (pages 43, 47, and 52)

[99] P. Zhan, S. Wang, J. Wang, L. Qu, K. Wang, Y. Hu, and X. Li, “Temporal anomaly detection on iiot-enabled manufacturing,” *Journal of Intelligent Manufacturing*, pp. 1–10, 2021. (pages 43 and 52)

[100] F. De Vita, D. Bruneo, and S. K. Das, “On the use of a full stack hardware/software infrastructure for sensor data fusion and fault prediction in industry 4.0,” *Pattern Recognition Letters*, vol. 138, pp. 30–37, 2020. (pages 43 and 52)

[101] C. Wang, “Iot anomaly detection method in intelligent manufacturing industry based on trusted evaluation,” *The International Journal of Advanced Manufacturing Technology*, vol. 107, no. 3, pp. 993–1005, 2020. (pages 43 and 52)

[102] Y. Peng, A. Tan, J. Wu, and Y. Bi, “Hierarchical edge computing: A novel multi-source multi-dimensional data anomaly detection scheme for industrial internet of things,” *IEEE Access*, vol. 7, pp. 111257–111270, 2019. (pages 43 and 52)

[103] S. Madhawa, P. Balakrishnan, and U. Arumugam, “Employing invariants for anomaly detection in software defined networking based industrial internet of things,” *Journal of Intelligent & Fuzzy Systems*, vol. 35, no. 2, pp. 1267–1279, 2018. (pages 43 and 52)

[104] V. Rousopoulou, T. Vafeiadis, A. Nizamis, I. Iakovidis, L. Samaras, A. Kirtsoglou, K. Georgiadis, D. Ioannidis, and D. Tzovaras, "Cognitive analytics platform with ai solutions for anomaly detection," *Computers in Industry*, vol. 134, p. 103555, 2022. (pages 44 and 52)

[105] A. Kumar, M. Shridhar, S. Swaminathan, and T. J. Lim, "Machine learning-based early detection of iot botnets using network-edge traffic," *Computers & Security*, p. 102693, 2022. (pages 44, 45, 47, and 52)

[106] M. S. S. Garmaroodi, F. Farivar, M. S. Haghighi, M. A. Shoorehdeli, and A. Jolfaei, "Detection of anomalies in industrial iot systems by data mining: Study of christ osmotron water purification system," *IEEE Internet of Things Journal*, 2020. (pages 44 and 52)

[107] I. Razzak, K. Zafar, M. Imran, and G. Xu, "Randomized nonlinear one-class support vector machines with bounded loss function to detect of outliers for large scale iot data," *Future Generation Computer Systems*, vol. 112, pp. 715–723, 2020. (pages 44 and 52)

[108] H. Yang, S. Liang, J. Ni, H. Li, and X. S. Shen, "Secure and efficient k nn classification for industrial internet of things," *IEEE Internet of Things Journal*, vol. 7, no. 11, pp. 10945–10954, 2020. (pages 44 and 52)

[109] Y. Shi, F. Li, W. Song, X.-Y. Li, and J. Ye, "Energy audition based cyber-physical attack detection system in iot," in *Proceedings of the ACM Turing Celebration Conference-China*, pp. 1–5, 2019. (pages 44, 45, and 52)

[110] R. Bodo, M. Bertocco, and A. Bianchi, "Feature ranking under industrial constraints in continuous monitoring applications based on machine learning techniques," in *2020 IEEE International Instrumentation and Measurement Technology Conference (I2MTC)*, pp. 1–6, IEEE, 2020. (pages 44 and 52)

[111] L. A. C. Ahakonye, C. I. Nwakanma, J. M. Lee, and D.-S. Kim, "Agnostic chdt technique for scada network high-dimensional data-aware intrusion detection system," *IEEE Internet of Things Journal*, 2023. (page 44)

[112] J.-f. Cui, H. Xia, R. Zhang, B.-x. Hu, and X.-g. Cheng, "Optimization scheme for intrusion detection scheme gbdt in edge computing center," *Computer Communications*, vol. 168, pp. 136–145, 2021. (pages 44 and 52)

[113] M. Douiba, S. Benkirane, A. Guezzaz, and M. Azrour, "An improved anomaly detection model for iot security using decision tree and gradient boosting," *The Journal of Supercomputing*, vol. 79, no. 3, pp. 3392–3411, 2023. (pages 44, 45, and 52)

[114] Y. Yang, X. Yang, M. Heidari, M. A. Khan, G. Srivastava, M. Khosravi, and L. Qi, "Astream: Data-stream-driven scalable anomaly detection with accuracy guarantee in iiot environment," *IEEE Transactions on Network Science and Engineering*, 2022. (pages 44 and 52)

[115] M. Elnour, N. Meskin, K. Khan, and R. Jain, "Application of data-driven attack detection framework for secure operation in smart buildings," *Sustainable Cities and Society*, vol. 69, p. 102816, 2021. (pages 44 and 52)

[116] S. Garg, K. Kaur, S. Batra, G. Kaddoum, N. Kumar, and A. Boukerche, "A multi-stage anomaly detection scheme for augmenting the security in iot-enabled applications," *Future Generation Computer Systems*, vol. 104, pp. 105–118, 2020. (pages 44 and 52)

[117] L. A. C. Ahakonye, C. I. Nwakanma, J.-M. Lee, and D.-S. Kim, "Scada intrusion detection scheme exploiting the fusion of modified decision tree and chi-square feature selection," *Internet of Things*, vol. 21, p. 100676, 2023. (pages 44, 45, and 52)

[118] J. Su, S. He, and Y. Wu, "Features selection and prediction for iot attacks," *High-Confidence Computing*, vol. 2, no. 2, p. 100047, 2022. (pages 44 and 52)

[119] V. Rey, P. M. S. Sánchez, A. H. Celdrán, and G. Bovet, "Federated learning for malware detection in iot devices," *Computer Networks*, p. 108693, 2022. (pages 44 and 52)

[120] J. He, L. Kong, T. Frondelius, O. Silvén, and M. Juntti, "Decision triggered data transmission and collection in industrial internet of things," in *2020 IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 1–5, IEEE, 2020. (pages 44, 45, and 52)

[121] X. Zhang, J. Li, D. Zhang, J. Gao, and H. Jiang, "Research on feature selection for cyber attack detection in industrial internet of things," in *Proceedings of the 2020 International Conference on Cyberspace Innovation of Advanced Technologies*, pp. 256–262, 2020. (pages 44 and 52)

[122] S. D. D. Anton, A. P. Lohfink, C. Garth, and H. D. Schotten, "Security in process: Detecting attacks in industrial process data," in *Proceedings of the Third Central European Cybersecurity Conference*, pp. 1–6, 2019. (pages 44 and 52)

[123] D. Raposo, A. Rodrigues, S. Sinche, J. S. Silva, and F. Boavida, "Security and fault detection in in-node components of iiot constrained devices," in *2019 IEEE 44th Conference on Local Computer Networks (LCN)*, pp. 282–290, IEEE, 2019. (pages 44, 52, and 54)

[124] D. Raposo, A. Rodrigues, S. Sinche, J. S. Silva, and F. Boavida, "Securing wirelesshart: Monitoring, exploring and detecting new vulnerabilities," in *2018 IEEE 17th International Symposium on Network Computing and Applications (NCA)*, pp. 1–9, IEEE, 2018. (pages 44 and 52)

[125] Z. Ouyang, X. Sun, J. Chen, D. Yue, and T. Zhang, "Multi-view stacking ensemble for power consumption anomaly detection in the context of industrial internet of things," *IEEE Access*, vol. 6, pp. 9623–9631, 2018. (pages 44 and 52)

[126] D. Nedeljkovic and Z. Jakovljevic, "Cnn based method for the development of cyber-attacks detection algorithms in industrial control systems," *Computers & Security*, vol. 114, p. 102585, 2022. (pages 45, 46, and 52)

[127] Y. Liu, T. Zhi, M. Shen, L. Wang, Y. Li, and M. Wan, "Software-defined ddos detection with information entropy analysis and optimized deep learning," *Future Generation Computer Systems*, vol. 129, pp. 99–114, 2022. (pages 46 and 52)

[128] C.-B. Seo, G. Lee, Y. Lee, and S.-H. Seo, "Echo-guard: Acoustic-based anomaly detection system for smart manufacturing environments," in *International Conference on Information Security Applications*, pp. 64–75, Springer, 2021. (pages 45, 46, and 52)

[129] S. H. Park, H. J. Park, and Y.-J. Choi, "Rnn-based prediction for network intrusion detection," in *2020 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*, pp. 572–574, IEEE, 2020. (pages 45, 46, and 52)

[130] Y. Wang, M. Perry, D. Whitlock, and J. W. Sutherland, "Detecting anomalies in time series data from a manufacturing system using recurrent neural networks," *Journal of Manufacturing Systems*, 2020. (pages 45, 46, 47, and 52)

[131] H. Wang, S. Mumtaz, H. Li, J. Liu, and F. Yang, "An identification strategy for unknown attack through the joint learning of space–time features," *Future Generation Computer Systems*, vol. 117, pp. 145–154, 2021. (pages 45, 46, and 52)

[132] F. Kong, J. Li, B. Jiang, H. Wang, and H. Song, "Integrated generative model for industrial anomaly detection via bi-directional lstm and attention mechanism," *IEEE Transactions on Industrial Informatics*, 2021. (pages 46 and 52)

[133] D. Wu, Z. Jiang, X. Xie, X. Wei, W. Yu, and R. Li, "Lstm learning with bayesian and gaussian processing for anomaly detection in industrial iot," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 8, pp. 5244–5253, 2019. (pages 46, 47, and 52)

[134] D. Li, N. Gebraeel, and K. Paynabar, "Detection and differentiation of replay attack and equipment faults in scada systems," *IEEE Transactions on Automation Science and Engineering*, vol. 18, no. 4, pp. 1626–1639, 2020. (pages 45, 46, 47, 50, and 52)

[135] T. T. Huong, T. P. Bac, D. M. Long, T. D. Luong, N. M. Dan, B. D. Thang, K. P. Tran, *et al.*, "Detecting cyberattacks using anomaly detection in industrial control systems: A federated learning approach," *Computers in Industry*, vol. 132, p. 103509, 2021. (pages 45, 46, and 52)

[136] M. Savic, M. Lukic, D. Danilovic, Z. Bodroski, D. Bajović, I. Mezei, D. Vukobratovic, S. Skrbic, and D. Jakovetić, "Deep learning anomaly detection for cellular iot with applications in smart logistics," *IEEE Access*, vol. 9, pp. 59406–59419, 2021. (pages 46 and 52)

[137] G. Bernieri, M. Conti, and F. Turrin, "Kingfisher: An industrial security framework based on variational autoencoders," in *Proceedings of the 1st Workshop on Machine Learning on Edge in Sensor Systems*, pp. 7–12, 2019. (pages 46 and 52)

[138] M. Al-Hawawreh and E. Sitnikova, "Industrial internet of things based ransomware detection using stacked variational neural network," in *Proceedings of the 3rd International Conference on Big Data and Internet of Things*, pp. 126–130, 2019. (pages 45, 46, and 52)

[139] Y. Liu, N. Kumar, Z. Xiong, W. Y. B. Lim, J. Kang, and D. Niyato, "Communication-efficient federated learning for anomaly detection in industrial internet of things," in *GLOBECOM 2020-2020 IEEE Global Communications Conference*, pp. 1–6, IEEE, 2020. (pages 45, 46, and 52)

[140] I. A. Khan, N. Moustafa, D. Pi, K. M. Sallam, A. Y. Zomaya, and B. Li, "A new explainable deep learning framework for cyber threat discovery in industrial iot networks," *IEEE Internet of Things Journal*, 2021. (pages 45, 46, and 52)

[141] D. Mukherjee, "A novel strategy for locational detection of false data injection attack," *Sustainable Energy, Grids and Networks*, p. 100702, 2022. (pages 45, 46, and 52)

[142] Z. Chen, D. Chen, X. Zhang, Z. Yuan, and X. Cheng, "Learning graph structures with transformer for multivariate time-series anomaly detection in iot," *IEEE Internet of Things Journal*, vol. 9, no. 12, pp. 9179–9189, 2021. (pages 45, 46, 47, and 52)

[143] R. Kozik, M. Pawlicki, and M. Choraś, "A new method of hybrid time window embedding with transformer-based traffic data classification in iot-networked environment," *Pattern Analysis and Applications*, vol. 24, no. 4, pp. 1441–1449, 2021. (pages 45, 46, 47, and 52)

[144] S.-J. Wang, C. X. Cai, Y.-W. Tseng, and K. S.-M. Li, "Feature selection for malicious traffic detection with machine learning," in *2020 International Computer Symposium (ICS)*, pp. 414–419, IEEE, 2020. (page 45)

[145] M. Al-Hawawreh, E. Sitnikova, and F. den Hartog, "An efficient intrusion detection model for edge system in brownfield industrial internet of things," in *Proceedings of the 3rd International Conference on Big Data and Internet of Things*, pp. 83–87, 2019. (pages 46 and 52)

[146] K. S. Sankaran and B.-H. Kim, "Deep learning based energy efficient optimal rmc-cnn model for secured data transmission and anomaly detection in industrial iot," *Sustainable Energy Technologies and Assessments*, vol. 56, p. 102983, 2023. (pages 45, 46, 47, and 52)

[147] M. Dzaferagic, N. Marchetti, and I. Macaluso, "Fault detection and classification in industrial iot in case of missing sensor data," 2021. (pages 45, 46, and 52)

[148] M. Priyadarsini and N. Sonekar, "A cnn-based approach for anomaly detection in smart grid systems," *Electric Power Systems Research*, vol. 238, p. 111077, 2025. (page 46)

[149] Y. K. Saheed, A. I. Omole, and M. O. Sabit, "Ga-madam-iiot: A new lightweight threats detection in the industrial iot via genetic algorithm with attention mechanism and lstm on multivariate time series sensor data," *Sensors International*, p. 100297, 2024. (page 46)

[150] D. He, H. Wang, T. Deng, J. Liu, and J. Wang, "Improving iiot security: Unveiling threats through advanced side-channel analysis," *Computers & Security*, p. 104135, 2024. (page 46)

[151] Y. Abudurexiti, G. Han, F. Zhang, and L. Liu, "An explainable unsupervised anomaly detection framework for industrial internet of things," *Computers & Security*, p. 104130, 2024. (page 46)

[152] Y. Fan, T. Fu, N. I. Listopad, P. Liu, S. Garg, and M. M. Hassan, "Utilizing correlation in space and time: Anomaly detection for industrial internet of things (iiot) via spatiotemporal gated graph attention network," *Alexandria Engineering Journal*, vol. 106, pp. 560–570, 2024. (page 46)

[153] S. M. S. Bukhari, M. H. Zafar, M. Abou Houran, Z. Qadir, S. K. R. Moosavi, and F. Sanfilippo, "Enhancing cybersecurity in edge iiot networks: An asynchronous federated learning approach with a deep hybrid detection model," *Internet of Things*, p. 101252, 2024. (page 46)

[154] Y. Chang, J. Chen, R. Su, J. Xie, and A. Li, "Two-phase dual-adversarial agents with multivariate information for unsupervised anomaly detection of iiot-edge devices," *IEEE Internet of Things Journal*, 2024. (page 46)

[155] Y.-C. Yu, Y.-C. Ouyang, L.-W. Wu, C.-A. Lin, and K.-Y. Tsai, "Multivariate time-series anomaly detection in iot with a bi-dual gm gru autoencoder," in *2024 IEEE 48th Annual Computers, Software, and Applications Conference (COMPSAC)*, pp. 746–754, IEEE, 2024. (page 46)

[156] S. Halder and T. Newe, "Radio fingerprinting for anomaly detection using federated learning in lora-enabled industrial internet of things," *Future Generation Computer Systems*, 2023. (pages 46 and 52)

[157] A. S. Kumar, S. Raja, N. Pritha, H. Raviraj, R. B. Lincy, and J. J. Rubia, "An adaptive transformer model for anomaly detection in wireless sensor networks in real-time," *Measurement: Sensors*, vol. 25, p. 100625, 2023. (pages 46 and 52)

[158] A. Ba, F. Lorenzi, and J. Ploennigs, "Monitoring of iot systems at the edges with transformer-based graph convolutional neural networks," in *2022 IEEE International Conference on Edge Computing and Communications (EDGE)*, pp. 41–49, IEEE, 2022. (pages 46, 47, and 52)

[159] J. Hu, K. Kaur, H. Lin, X. Wang, M. M. Hassan, I. Razzak, and M. Hammoudeh, "Intelligent anomaly detection of trajectories for iot empowered maritime transportation systems," *IEEE Transactions on Intelligent Transportation Systems*, 2022. (pages 46, 47, and 52)

[160] J. Pan, W. Ji, B. Zhong, P. Wang, X. Wang, and J. Chen, "Duma: Dual mask for multivariate time series anomaly detection," *IEEE Sensors Journal*, 2022. (pages 46 and 52)

[161] Y. Feng, J. Chen, Z. Liu, H. Lv, and J. Wang, "Full graph autoencoder for one-class group anomaly detection of iiot system," *IEEE Internet of Things Journal*, vol. 9, no. 21, pp. 21886–21898, 2022. (pages 46 and 52)

[162] B. Weinger, J. Kim, A. Sim, M. Nakashima, N. Moustafa, and K. J. Wu, "Enhancing iot anomaly detection performance for federated learning," *Digital Communications and Networks*, 2022. (pages 46 and 52)

[163] O. Friha, M. A. Ferrag, L. Shu, L. Maglaras, K.-K. R. Choo, and M. Nafaa, "Felids: Federated learning-based intrusion detection system for agricultural internet of things," *Journal of Parallel and Distributed Computing*, 2022. (pages 46 and 52)

[164] K. Yang, Y. Shi, Z. Yu, Q. Yang, A. K. Sangaiah, and H. Zeng, "Stacked one-class broad learning system for intrusion detection in industry 4.0," *IEEE Transactions on Industrial Informatics*, 2022. (pages 46 and 52)

[165] W. Wangwang, Z. Yunchun, L. Chengjie, S. Xuchenming, Z. Yuting, and Z. Xin, "Network traffic oriented malware detection in iot (internet-of-things)," in *2021 International Conference on Networking and Network Applications (NaNA)*, pp. 301–307, IEEE, 2021. (pages 46 and 52)

[166] X. Wang, S. Garg, H. Lin, J. Hu, G. Kaddoum, M. J. Piran, and M. S. Hossain, "Towards accurate anomaly detection in industrial internet-of-things using hierarchical federated learning," *IEEE Internet of Things Journal*, 2021. (pages 46, 47, and 52)

[167] V. Ketonen and J. O. Blech, "Anomaly detection for injection molding using probabilistic deep learning," in *2021 4th IEEE International Conference on Industrial Cyber-Physical Systems (ICPS)*, pp. 70–77, IEEE, 2021. (pages 46 and 52)

[168] Y. Liu, S. Garg, J. Nie, Y. Zhang, Z. Xiong, J. Kang, and M. S. Hossain, "Deep anomaly detection for time-series data in industrial iot: a communication-efficient on-device federated learning approach," *IEEE Internet of Things Journal*, vol. 8, no. 8, pp. 6348–6358, 2020. (pages 46, 47, and 52)

[169] Y. Wu, H.-N. Dai, and H. Tang, "Graph neural networks for anomaly detection in industrial internet of things," *IEEE Internet of Things Journal*, 2021. (pages 46 and 52)

[170] X. Zhou, Y. Hu, W. Liang, J. Ma, and Q. Jin, "Variational lstm enhanced anomaly detection for industrial big data," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 5, pp. 3469–3477, 2020. (pages 46 and 52)

[171] Y. Li, Y. Xu, Z. Liu, H. Hou, Y. Zheng, Y. Xin, Y. Zhao, and L. Cui, "Robust detection for network intrusion of industrial iot based on multi-cnn fusion," *Measurement*, vol. 154, p. 107450, 2020. (pages 46 and 52)

[172] K. Demertzis, L. Iliadis, N. Tziritas, and P. Kikiras, "Anomaly detection via blockchained deep learning smart contracts in industry 4.0," *Neural Computing and Applications*, vol. 32, no. 23, pp. 17361–17378, 2020. (pages 46 and 52)

[173] F. De Vita, D. Bruneo, and S. K. Das, "A novel data collection framework for telemetry and anomaly detection in industrial iot systems," in *2020 IEEE/ACM Fifth International Conference on Internet-of-Things Design and Implementation (IoTDI)*, pp. 245–251, IEEE, 2020. (pages 46, 47, and 52)

[174] P. Ferrari, S. Rinaldi, E. Sisinni, F. Colombo, F. Ghelfi, D. Maffei, and M. Malara, "Performance evaluation of full-cloud and edge-cloud architectures for industrial iot anomaly detection based on deep learning," in *2019 II Workshop on Metrology for Industry 4.0 and IoT (MetroInd4. 0&IoT)*, pp. 420–425, IEEE, 2019. (pages 46, 47, and 52)

[175] S. Liu, X. Chen, X. Peng, and R. Xiao, "Network log anomaly detection based on gru and svdd," in *2019 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/-SustainCom)*, pp. 1244–1249, IEEE, 2019. (pages 46, 47, and 52)

[176] V. Krundyshev and M. Kalinin, "Hybrid neural network framework for detection of cyber attacks at smart infrastructures," in *Proceedings of the 12th International Conference on Security of Information and Networks*, pp. 1–7, 2019. (pages 46 and 52)

[177] G. Bae, S. Jang, M. Kim, and I. Joe, "Autoencoder-based on anomaly detection with intrusion scoring for smart factory environments," in *International Conference on Parallel and Distributed Computing: Applications and Technologies*, pp. 414–423, Springer, 2018. (pages 46 and 52)

[178] D. Kim, H. Yang, M. Chung, S. Cho, H. Kim, M. Kim, K. Kim, and E. Kim, "Squeezed convolutional variational autoencoder for unsupervised anomaly detection in edge device industrial internet of things," in *2018 international conference on information and computer technologies (icict)*, pp. 67–71, IEEE, 2018. (pages 46, 47, and 52)

[179] A.-H. Muna, N. Moustafa, and E. Sitnikova, "Identification of malicious activities in industrial internet of things based on deep learning models," *Journal of Information security and applications*, vol. 41, pp. 1–11, 2018. (pages 46 and 52)

[180] P. Schneider and K. Böttinger, "High-performance unsupervised anomaly detection for cyber-physical system networks," in *Proceedings of the 2018 workshop on cyber-physical systems security and privacy*, pp. 1–12, 2018. (pages 46, 47, and 52)

[181] N. Tandiya, A. Jauhar, V. Marojevic, and J. H. Reed, "Deep predictive coding neural network for rf anomaly detection in wireless networks," in *2018 IEEE International Conference on Communications Workshops (ICC Workshops)*, pp. 1–6, IEEE, 2018. (pages 46 and 52)

[182] X. Wang, S. Garg, H. Lin, J. Hu, G. Kaddoum, M. J. Piran, and M. S. Hossain, "Toward accurate anomaly detection in industrial internet of things using hierarchical federated learning," *IEEE Internet of Things Journal*, vol. 9, no. 10, pp. 7110–7119, 2021. (pages 49 and 52)

[183] M. Taheri, K. Khorasani, I. Shames, and N. Meskin, "Cyberattack and machine-induced fault detection and isolation methodologies for cyber-physical systems," *IEEE Transactions on Control Systems Technology*, 2023. (page 50)

[184] Q. Abu Al-Haija and S. Zein-Sabatto, "An efficient deep-learning-based detection and classification system for cyber-attacks in iot communication networks," *Electronics*, vol. 9, no. 12, p. 2152, 2020. (page 50)

[185] Q. Abu Al-Haija and M. Al-Dala'ien, "Elba-iot: An ensemble learning model for botnet attack detection in iot networks," *Journal of Sensor and Actuator Networks*, vol. 11, no. 1, p. 18, 2022. (page 50)

[186] Q. Abu Al-Haija, A. Al Badawi, and G. R. Bojja, "Boost-defence for resilient iot networks: A head-to-toe approach," *Expert Systems*, vol. 39, no. 10, p. e12934, 2022. (page 50)

[187] K. Albulayhi, Q. Abu Al-Haija, S. A. Alsuhibany, A. A. Jillepalli, M. Ashrafuzzaman, and F. T. Sheldon, "Iot intrusion detection using machine learning with a novel high performing feature selection method," *Applied Sciences*, vol. 12, no. 10, p. 5015, 2022. (page 50)

[188] G. Tertytchny, N. Nicolaou, and M. K. Michael, "Classifying network abnormalities into faults and attacks in iot-based cyber physical systems using machine learning," *Microprocessors and Microsystems*, vol. 77, p. 103121, 2020. (page 51)

[189] S. Gill, B. Lee, and E. Neto, "Context aware model-based cleaning of data streams," in *2015 26th Irish Signals and Systems Conference (ISSC)*, pp. 1–6, IEEE, 2015. (page 52)

[190] X. Berger, "Rpi monitor," 2022. `https://github.com/XavierBerger/RPi-Monitor`. (page 82)

[191] A. A. Mantha, A. Hussain, and G. Ravikumar, "Hil testbed-based auto feature extraction and data generation framework for ml/dl-based anomaly detection and classification," in *2024 IEEE Power & Energy Society Innovative Smart Grid Technologies Conference (ISGT)*, pp. 1–5, IEEE, 2024. (page 109)

[192] M. Rodriguez, "Iiot anomaly classification repository," 2024. `https://github.com/mluciarodriguez/iiot_anomaly_classification`. (page 109)