Università degli Studi di Firenze
Dipartimento di Ingegneria dell'Informazione (DINFO)
Corso di Dottorato in Ingegneria dell'Informazione
Curriculum: Informatica

Machine Learning application to IoT and IIoT security and reliability

Candidate
Chiara Camerota

Supervisors
Prof. Tommaso Pecorella

Prof. Andrew D. Bagdanov

PhD Coordinator
Prof. Fabio Schoen

ciclo XXXVII, 2020-2023

Università degli Studi di Firenze, Dipartimento di Ingegneria
dell'Informazione (DINFO).

Thesis submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Information Engineering.

A Zaira

Acknowledgments

I would like to acknowledge the efforts and input of my supervisors, Prof. Tommaso Pecorella and Prof. Andrew D. Bagdanov, and all my colleagues at the DaCoNetS and LESC laboratories. I would like to thank also Dr. Flavio Esposito for his kind support and hospitality during my stay at St. Louis University.

Contents

Chapter 1

Introduction

The Internet of Things (IoT) is a network of interconnected devices and objects spanning various domains, including smart homes and industrial environments. Machine learning (ML) technologies often enhance this ecosystem, facilitating transformative potential through automation, real-time data processing, and efficient resource management. By optimizing processes, ML algorithms provide solutions that address data management, predictive analytics, anomaly detection, and cybersecurity. In industrial contexts, ML contributes to predictive maintenance and operational efficiency, while in networking, it enhances traffic analysis and threat detection. The application of ML enables IoT systems to adapt to evolving challenges, thereby improving resilience.

However, the implementation of these technologies presents challenges. Hardware and software limitations frequently vary by application field, and the successful adoption of standard data models depends on widespread acceptance. Moreover, the proposed ML approaches are typically tested offline, which may not fully represent the complexities of real-world scenarios. This thesis investigates effective machine learning techniques within industrial and networking contexts to improve data management and enhance security by analyzing large datasets and adapting to changing patterns. Emphasis is placed on the necessity of data model standards for efficient data flow management, interoperability, and scalability. Additionally, the focus is directed toward deploying machine learning-based solutions for malware detection, highlighting the importance of addressing data constraints.

Contributions. I introduce a novel tool that facilitates the seamless integration of new devices into existing networks, enabling a scalable and adaptable IoT infrastructure that supports diverse and dynamic devices. Building on this infrastructure, I employ Federated Learning as a decentralized approach to malware detection, leveraging the distributed nature of IoT to enhance security across devices while reducing variability in threat identification. To optimize these methods, I propose an approach for determining the minimum sample size needed for multi-class classification, ensuring efficient resource use without compromising classification performance. Additionally, I develop a Diffusion Probabilistic Model to generate synthetic traffic-based images for data augmentation, providing robust support for traffic analysis in cybersecurity contexts through enhanced data diversity. The robustness of this model will instill confidence in its ability to enhance data diversity. Together, these contributions create an integrated framework that addresses core challenges in IoT systems, significantly improving their reliability and resilience for real-world applications.

Limitations. In this work there are some limitations that are worth mention. The adoption of standard data models may prove ineffective without widespread acceptance within the industry. The proposed machine learning approaches depend on offline testing, which may not fully encompass the complexities of real-world scenarios. Additionally, concentrating on specific deployment aspects, such as prioritizing model stability over accuracy, may restrict broader applicability. Future work should address these limitations by exploring diverse deployment scenarios, conducting real-time testing, and refining techniques to enhance adaptability and scalability across Internet of Things applications.

Structure. The structure of the thesis is as follows. In Chapter 2, the literature review covers IoT architecture, the importance of standardization, and the existing security malware, including the role of Machine Learning in risk mitigation. Then, Chapter 3 presents an automatic IoT device harvesting system using an open-source platform, detailing its architecture, device registration processes, and validation experiments. In Chapter 4, Federated learning is introduced as a decentralized solution for IoT malware detection, with a methodology involving data pre-processing, model selection, and performance evaluation. Chapter 5 also addresses attack classification in low-information environments, tackling challenges related to limited data availability, and proposes innovative solutions to enhance IoT security. Finally, the work is thoroughly validated through experiments, offering reassurance about the practicality and effectiveness of the proposed solutions for more secure, scalable, and efficient IoT systems. The collaboration works are briefly presented in Chapter 7 and comprehend a study on Active Magnetic Bearings and fault recognitions. The conclusions are presented in Chapter 8.

Chapter 2

Literature review

With the increasing ubiquity of smart devices in our daily lives, the need for a new app is becoming a norm each time we connect to a new device. Statista's research department predicts that a staggering 75 billion devices will be connected to the Internet by 2025. However, the need for a unified control dashboard for managing these devices poses the risk of downloading unsafe apps. In essence, installing a new device is not only clunky but also opens the door to potential data theft by malicious users.
The following chapter focuses on these points and is a gentle introduction to the Internet of Things (IoT). It explains the open challenges and issues, focusing on security and how to address them.
In the first section, pragmatic and theorized examples exploit the problem of the lack of standards inn standard is also IoT and suggest ways to address it. The Europea introduced as a possible open-source solution. The chapter's final section exploits machine learning's promising potential for IoT security in real-world applications. It introduces anomaly detection techniques and more advanced methods, such as data augmentation, instilling hope for the future of IoT security.

2.1 Internet of things

The most appropriate way to define the Internet of Things is by the ISO/IEC 20924:2024 vocabulary:

infrastructure of interconnected entities, people, systems and information resources together with services which processes and reacts to information from the physical world and virtual world [61].

The above definition can be adapted as an environment with a large number of heterogeneous connected objects using several connectivity technologies such as ZigBee, WIFI, near-field communication (NFC), etc [4]. This definition includes connected cars, intelligent assistants, connected medical devices, airplanes, personal computers, smartphones, smart refrigerators, etc [45].

Additionally to the devices, the whole IoT platform and infrastructure (cloud services, software, etc.) are also in scope. This kind of technology is strictly related to the topics of Industry 4.0 (the Fourth Industrial Revolution); hence, the term Industrial IoT arises here. It refers to connected devices in an industrial and production environment, and then it was defined as *a paradigm shift towards digitalized, integrated, and smart value chains enabling distributed decision-making in production by incorporating new cyber-physical technologies such as IoT* by the European Union Agency for Cybersecurity (ENISA) [75].

The increasing presence of IoT in everyday tasks and the rapid growth of related technologies are automatizing the world scenario to improve the quality of human lives. That also enables the Internet-connected devices to connect together for different applications [112]. Hence, these objects are often small devices with low computation capacity and no standard security system, and they could implement their own protocol. Consequently, they need some manipulation or tool to become part of a pre-existing environment and implement protection against malicious attacks. These criticisms became the center of interest and must be studied and resolved due to the increasing presence of IoT devices. Fortunately, many researchers contribute to designing and developing protocols for different IoT fields. Also, Secure authentication is provided for different nodes or devices in a hierarchical IoT network. The following subsections exploit the IoT layers to better understand this work's contribution and the complex nature of the IoT's work.

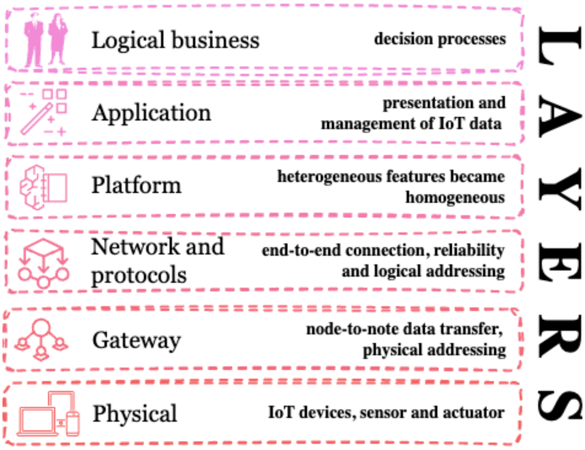

Logical business
decision processes
Application
presentation and management of IoT data
Platform
heterogeneous features became homogeneous
Network and protocols
end-to-end connection, reliability and logical addressing
Gateway
node-to-note data transfer, physical addressing
Physical
IoT devices, sensor and actuator
LAYERS

Figure 2.1: A generic representation of the whole IoT Stack, from the single device to the business layer, where the decisions are taken.

2.1.1 IoT stack

Based on the Service-Oriented Architecture (SOA) model, I propose the following stack models dedicated to the Internet of Things (IoT) ecosystem, spanning from the physical layer to the logic and business processes. This model integrates the technology, equipment, and components necessary for an IoT solution to function effectively, merging certain layers of the SOA while excluding others. It comprises seven layers presented in 2.1.

The physical layer gathers device data and converts it into digital signals using various sensing technologies such as Radio Frequency Identification (RFID), WSN, and the Global Positioning System [30]. Nanotechnologies and embedded intelligence can equip the devices with essential computing power. This allows the sensors and actuators to meet the operational requirements of the applications.

The gateway layer transfers data (as packets) between hosts connected to the physical layer. It handles packet boundary delineation, frame synchronization, sender and recipient addresses management, error detection in the physical media channel, and collision prevention [31]. It includes all network equipment, such as switches and routers, necessary for proper Third Generation (3G), Fourth Generation (4G), Fifth Generation (5G), Wi-Fi, infrared technology, ZigBee, and communication and routing protocols.

The network and protocol layer facilitates end-to-end reliability and communication across multiple networks. It ensures the correct order of data packet delivery, mitigates network congestion and multiplexes, maintains data integrity, and guarantees the reliability of transmitted data. This layer is crucial for system security, as it manages platform data and must ensure user safety [80]. Its logic parallels that of the data link layer, differing primarily in its ultimate goal: ensuring reliable data delivery between directly connected devices within a local network segment.

Once the data are in the systems, they have to be stored and ready to be analyzed. This is the role of the platform layer, which harvests all different devices and brokers, making the data homogeneous concerning a protocol or standard [43]. Ideally, This layer can recognize devices' interfaces, collect these data, and give them as input to the application layer.

The application layer is fundamental to all operations of IoT data; without it, the user cannot read and analyze the data. It is the front end of IoT architectures, and it provides the interfaces and tools needed to build IoT applications, such as those for smart homes, intelligent transportation, smart health, etc. The application layer can have different protocols; the most used are Message Queue Telemetry Transport (MQTT), Constrained Application Protocol (CoAP), Advanced Message Queuing Protocol (AMQP), HTTP, and Extensible Messaging and Presence Protocol (XMPP) [67].

The final logical business layer is not commonly explicit but includes the interaction between humans and things. It concerns users who need to make decisions or take other actions based on the data collected from various devices and apps.

The complexity of this type of ecosystem and the challenge of integrating different devices with different protocols or models make it necessary to study and analyze the platforms. The next subsefollowingn discusses this topic and provides a technical explanation of what a standard is and what it signifies in this environment.

2.1.2 Why do we need standards?

The standard plays a crucial, valuable, and user-friendly role in the Internet of Things experience. As we saw before, a device can adopt several protocols; often, different tasks have different optimal protocols. As the number of devices from various manufacturers continues to grow, ensuring that they cak together reliably, at scale, and securely becomes increasingly challenging. Standards play a crucial role in addressing these challenges by defining how devices communicate, exchange data, and operate within the larger ecosystem. [91].

Platforms such as FIWARE, with their open-source approach and standardized components, provide an ideal framework for creating intelligent applications that meet these requirements. This framework has been evolving for nearly a decade with substantial support from the European Commission. This support has been materialized through funding several projects aimed at both the development and enhancement of the platform (e.g., FI-WARE, FI-Core, FI-Next, SmartSDK) and the deployment of pilot projects (e.g., City Platform as a Service - CPaaS.io, frontierCities, Fiware4Water).

Step back to what a standard is; a good concept to start with is interoperability. This means that a device should be able to communicate effectively within a network, even if it has different hardware and software architectures. Interoperability is crucial for IoT, and standards like FIWARE's NGSI API ensure that data can be shared and utilized across various platforms and devices, regardless of origin [69]. Furthermore, a variable number of devices are part of the network, so scalability is important. Standards ensure that systems can expand without significant reengineering efforts [53]. For instance, FIWARE's standardized data models and APIs enable developers to easily integrate new devices and applications without compatibility concerns, allowing systems to grow seamlessly. The ease of use and multiple access should not be mistaken for a lack of safety. Security is extremely important, especially when IoT is used in a healthcare system and handles sensitive data. This chapter later explains that standardized security protocols are crucial to prevent vulnerabilities and protect IoT ecosystems from cyber threats. This is also important regarding reliability, particularly in mission-critical applications such as healthcare or industrial automation. Standards establish performance benchmarks, ensuring that devices and systems meet specific reliability criteria [16]. This provides that the information shared between devices is precise and reliable, which is essential for preserving the security of IoT networks. Uniform encryption, authentication, and data transmission standards, such as those implemented within FIWARE, reduce the likelihood of security breaches.

From an industrial perspective, there are other important considerations, such as the cost efficiency of maintaining the ecosystem and ensuring data consistency. Maintaining consistency in how the data is formatted, stored, and interpreted is crucial, especially in a world where various sensors and devices generate data. For example, by adopting open data standards like FIWARE, developers can reduce costs using reusable components and protocols. This, in turn, reduces the time and resources needed for integration, maintenance, and updates, resulting in more cost-effective IoT solutions. It also ensures that data from different sources can be interpreted uniformly, enabling more accurate and reliable analysis. In conclusion, in an increasingly interconnected world, the success of IoT systems depends on the ability of different devices to communicate, scale, and operate securely. We see how FIWARE promotes interoperability, scalability, security, and reliability standards to ensure the long-term sustainability and growth of the IoT ecosystem. Integrating platforms like FIWARE with global IoT standards not only simplifies development and reduces costs but also drives widespread adoption, enabling the creation of smart, connected solutions across various industries. The following section focused only on security and how machine learning algorithms can help with this aspect.

2.2 Security in IoT

The proliferation of IoT has precipitated the emergence of substantial security vulnerabilities in both domestic and industrial contexts. In addition, the complex nature of the Internet of Things (IoT) makes it susceptible to a diverse array of attacks, with each architecture layer vulnerable to a distinct set of threats. To illustrate, the physical layer is susceptible to manipulation, data falsification, and physical assaults [117]. Conversely, the network layer is vulnerable to compromises such as man-in-the-middle attacks, denial-of-service (DoS) attacks, and data interception [63]. Furthermore, the application layer, which interacts with end-users, is particularly susceptible to attacks on data privacy, authentication weaknesses, and unauthorized control of devices [85]. The conventional security measures are badly suited to address the distinctive attributes of IoT systems, including constrained device resources, extensive deployments, and heterogeneous architectures [59]. For this reason, traditional security measures, such as robust encryption, need too much calculus power and memory to be efficient on the devices [6]. Additionally, the vast number of devices involved in IoT ecosystems complicates scalability [53], and the same is done by centralized security mechanisms, which still make the network busy [37]. The diversity of devices in terms of their hardware, software, and communication protocols presents compatibility challenges, as shown above, and makes it difficult to implement consistent security policies. Privacy is also a significant concern when using these technologies. IoT systems gather large amounts of personal and environmental data, which can create vulnerabilities if not securely managed [87]. Therefore, it's essential to ensure secure communication and data integrity across all layers to uphold the overall security of IoT systems.

Machine learning (ML) is a suitable solution to address these kinds of issues. It also allows for the detection of anomalies, adaptation to evolving attack patterns, and real-time protection of devices. Alongside ML, techniques like data augmentation and advanced malware detection models have proven to be effective tools for addressing threats in IoT systems. This section examines malware detection challenges in IoT security, focusing on ML techniques that enhance traditional security systems.

2.2.1 Malware detection

The field of malware detection is extensive and encompasses numerous case studies. Malware attacks increasingly threaten both the IoT and traditional systems [13].

The literature contains several strategies for addressing this issue, which are collected, explored, and classified due to the heterogeneity of the problem and the available solutions. Machine learning techniques can help to analyze dynamics and memory by generalizing tasks with relevant data. Nascita's study shows that using customized data for IoT to feed an ML model improves performance [78]. This is also supported by [64], who demonstrate how to mitigate damage to IoT devices by intelligently identifying both known and emerging IoT malware. They achieve this by analyzing various device properties, converting them into images, and using dynamic analysis. However, these studies often overlook important factors such as the high cost of collecting extensive datasets, the scarcity of available data, and the excessive traffic generated by these approaches. In this paragraph, these issues are discussed, and the most advanced solution is presented to highlight the challenges and considerations.

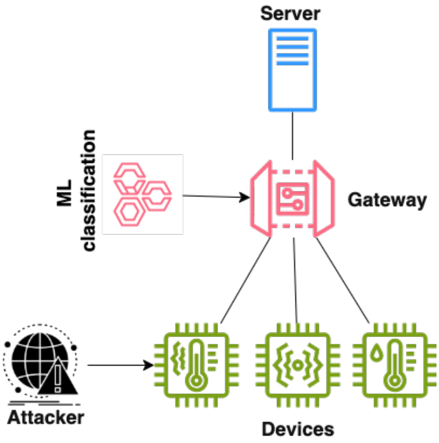


Figure 2.2: A simplified representation of IoT system. The devices are low-powered, so the ML classifier is installed on the gateway to enhance the environment's security.

Federated Learning

Figure 2.2 illustrates how the IoT ecosystem in a cloud environment, this integration presents a complex challenge, such as the privacy aspect or the more specific network issues [1]. To address these, a decentralized model should be considered over a centralized model to mitigate data exchange and communication channel availability issues. Federated Learning (FL) techniques offer a solution to this challenge by ensuring privacy [106]. These techniques enable machine learning model training on the device and only exchange parameters with the server without sharing the user's information.

An example of FL applied to an IoT network is [89]. The authors present a model where the federated approach prioritizes participant privacy while achieving results comparable to those of centralized models. The authors also investigate the safety and robustness of this approach and demonstrate that the baseline model aggregation averaging step is highly vulnerable to attacks, even with a single adversary. However, they do not consider the impact on the network or analyze the comparison regarding the results' variance.

Another reason to consider decentralized methods over centralized ones is the challenge of combining anonymized data from heterogeneous and differently shaped data sources. Anonymized data alone cannot ensure equal client distribution, and a centralized model may face difficulties in handling the non-independent and identically distributed (non-IID) nature of device data. As a result, this can lead to potential bias and reduced accuracy during model training. In this situation, clients may be unreliable and experience higher failure rates or dropouts due to their dependence on less robust communication media, such as Wi-Fi, and power-constrained systems, such as smartphones and IoT devices. Taheri et al. [109] suggest a novel federated learning approach to address privacy, robustness, and model training concerns when dealing with non-IID data. Still, they focus our work exclusively on the ML model aspects and do not consider the environment. For interested readers, [86] comprehensively analyzes various FL methodologies that apply to IoT systems. Additionally, reference [114] outlines potential research topics.

Adversarial training

Adversarial training is a powerful technique that leverages the concept of training a model by pitting it against another model. Here’s an example: one model (the generator) tries to create realistic forgeries (adversarial examples) that can deceive another model (the discriminator) into misclassifying them. In response, the discriminator constantly trains to improve its ability to detect these forgeries. This ongoing competition between the two models strengthens both. The discriminator becomes skilled at identifying subtle variations in the data that could otherwise confuse it in real-world scenarios. While powerful, adversarial training can be computationally expensive and complex to implement.

Let us now examine the potential applications of this technique in the context of malware detection. In their study, Shaukat and colleagues proposed a novel method that merges static and dynamic analysis for enhanced detection [97]. This approach involves visualizing malware as an image, extracting key features using deep learning, and then classifying it with machine learning. This not only eliminates the need for complex manual feature engineering but also achieves high accuracy on benchmark datasets. The authors additionally propose data augmentation techniques to address data scarcity and improve the generalizability of their method. Overall, this research offers a promising and efficient solution for the ongoing battle against malware.

Another illustrative example is provided in [52]. The authors introduce a novel approach that addresses the limitations of existing methods, which were unable to effectively handle the distinctive characteristics of malware data (e.g., its binary nature) and preserve the functionality of malware after manipulation. The study also examines potential defensive strategies inspired by computer vision, demonstrating the potential for mitigating these adversarial attacks.

Lack of data

The cornerstone of machine learning's success is its ability to extract knowledge from data and effectively apply that knowledge to entirely new situations. This learning process requires a delicate balancing act. Traditionally, there has been a direct correlation between the amount of data a model is exposed to and its performance. However, this is not a case of *bigger is always better*. Complex models, often characterized by a large number of parameters, can become overly dependent on the features of their training data, leading to a phenomenon known as overfitting. An overfit model performs well on data that is already known, but it fails to generalize to data that has not been seen before. In the state of the art, increasing the number of data points is a key strategy to combat overfitting. As shown in [122], with a greater number of data points, the model encounters a wider spectrum of patterns and learns a more robust representation of the fundamental relationships at play. This enables it to distinguish between features that are generally applicable and those that are specific to the training set.

However, there is no universal approach to determining sample size. Several factors must be considered, including the inherent complexity of the model. Models with fewer parameters and a simpler structure may perform well on smaller data sets, as evidenced in [51]. Conversely, complex models with a large number of parameters require significantly more data to avoid overfitting. Recent research suggests that surprisingly small data sets may be sufficient for certain tasks and architectures. Studies have demonstrated that with carefully selected models and appropriate training techniques, as few as 15,000 data points can be sufficient to achieve adequate performance [8]. Furthermore, in biological research, the lack of data is a significant issue, so it is straightforward to identify different methods to discover an appropriate sample size without encountering an overfitting problem. In [96], the authors investigate the impact of *separate sampling*, where data points for classifier construction are collected from two phenotypes with predetermined, non-random class sizes. Through simulations with synthetic and real data, the research identifies the detrimental effects of separate sampling, proposes a sample-based mini-max sampling ratio, and extends classical population-based mini-max sampling ratios to arbitrary distributions for improved classifier design. Another example is [41]; this paper discusses the critical influence of testing set size on accuracy assessment and comparison in classification evaluations. It highlights the potential impact of using inadequately sized samples and emphasizes the importance of statistical principles in sample size determination to control for Type II errors and ensure meaningful accuracy assessments and comparisons. It also provides guidelines for specifying effect size, significance level, power, and confidence limits to effectively design studies, avoid underpowered analyses, and improve the interpretation and richness of classification evaluation results.

The aforementioned techniques are not readily comprehensible or implementable. Consequently, augmenting the sample size represents a straightforward approach, although it is not always feasible. Data collection can be time-consuming, costly, or even impracticable in certain scenarios. In such instances, state-of-the-art techniques such as adversarial training and data augmentation can be employed. These methods offer ingenious ways to leverage existing data more effectively and improve model robustness, even with limited datasets.

Data Augmentation

The field of data augmentation is in a state of constant evolution. It involves the artificial expansion of the dataset through the application of random transformations to existing data points. Traditionally, these transformations can include simple geometric operations such as flips, rotations, and scaling, or more complex manipulations. By creating variations of existing data, this method helps the model become more invariant to minor changes in the input data. This, in turn, improves the model's ability to generalize to unseen examples. Data augmentation is particularly valuable in the context of limited datasets, as it introduces a degree of artificial diversity that the model can learn from.

In [26], the authors propose a method that combines Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTMs) networks to effectively handle the high variance in daily energy consumption caused by various external and internal factors and address a binary classification of malware traffic data, also employing traffic-based images. These approaches incorporate multiple data and metadata types to introduce redundancy in the dataset. These studies confirm how the best prediction results are strictly related to the amount and diversity of data in a domain where they are often unavailable. Given its importance, a method that explores how to achieve high-accuracy results while reducing the amount of collected data in IoT Malware detection was still missing. The literature suggests several valuable techniques for addressing this lack of data, and data augmentation represents a particularly effective and constantly evolving approach. An example is [11], where the authors utilize a machine learning method to categorize images based on their malware traffic patterns. The research introduces a novel convolutional neural network (CNN) design incorporating dilated convolution operations, channel squeezing, and boosting to detect IoT-specific malicious patterns. Additionally, the study applies a simple data augmentation method like image rotation, which proves to be less effective in this scenario, as the images do not have a natural directional orientation. A more interesting example that combines generative methods and other deep learning approaches to achieving high levels of accuracy is shown in [2].

While their findings indicate that GANs could improve detection performance, GAN-generated traffic images still produce a high false negative rate. In fact, the synthetic images closely resemble the distribution of the original samples and fail to introduce sufficient noise into the training set to achieve a better generalization.

Its authors investigated the potential of Generative Adversarial Networks (GANs) and Variational Autoencoders (VAEs) to improve malware traffic classification. These models can generate synthetic malware samples, which can be integrated into the training process. The findings suggest that GANs may be suitable for generating synthetic training data, potentially improving malware detection performance, despite their work showing that using GAN artificial images does not improve the baseline or the False Negative rate. That happens because these images do not introduce enough noise in the train set but generate a configuration similar to the original ones.

To achieve realistic traffic-based images and identify traffic profiles, the author of [65] introduced the stable diffusion model to augment data. They demonstrated how profile detection improves by changing the generative model and using a text-to-image model. Thus, their analyses require a double data set preparation: one for the RGB images and one for the stable diffusion model prompts. However, this remains an interesting example of how the denoising models can improve image generation, but the cost of the process should be considered. This leads to the idea of using the Denoising Diffusion Probabilistic Model (DDPM) as an image-to-image generative approach. This method provides a powerful tool for generating high-fidelity images and offers a more precise result in terms of Frèchet inception distance (FID) compared to GANs [56, 107].

Chapter 3

Automatic harvesting of IoT devices in an open sources platform

This chapter discusses the lack of standards in the Internet of Things (IoT), focusing on the case study of Snap4City. This open-source IoT platform, designed for Smart Cities and Industry 4.0, facilitates the integration of new brokers and devices into extensive ecosystems. I employ Snap4City's terminology to distinguish between internal brokers (those managed directly by the platform) and external brokers (those managed by third parties). The platform controls internal brokers, while third parties manage external brokers. Both types of brokers can provide multiple services (multi-service) and accommodate numerous tenants simultaneously (multi-tenant), thereby ensuring efficient resource management. They are designed to protect client confidentiality while maintaining data isolation in a multi-tenant environment. Additionally, these brokers can manage and process various data models, supporting various data structures and formats.

The interoperable management of these complex networks is a dynamic process that begins with the registration of IoT devices. This process is ongoing, mirroring the continuous evolution of IoT networks. To effectively manage all objects, Snap4City defines the IoT Directory as a collection of automated tools that

streamline the registration process. This directory harvests and organizes devices managed by external brokers' single or multi-tenant services. It also automates the registration and management of Data Models, including custom Data Models. [1] [2]

3.1 Introduction

The Internet of Things (IoT) defines a paradigm for the computation and communication among things that everyone uses more and more daily [47]. It is due to the intense deployment campaign worldwide about Low-Power Wide Area Network technologies [76]. Nowadays, the real world and IoT devices are integrated with some humans in the loop. Communication among devices may support various protocols (e.g., Message Queue Telemetry Transport or MQTT, Next Generation Service Interfaces or NGSI, Advanced Message Queuing Protocol or AMQP, Constrained Application Protocol or CoaP); thus, the cloud-fog infrastructures are exploited, as well as the management of the information [123]. In terms of data management, the complexity is growing, not only for the huge amount of data but also for the need for interoperability and abstraction. The **Gateway** concept is relevant to managing IoT devices into an **IoT Platform**. It may be integrated with one or several **IoT Brokers** to send/receive data to/from devices. The Gateways and its IoT Brokers are typically based on a single protocol and managed by third parties as a public service for several customers interested in the same area (for example, LoraWAN services [3]). A Gateway may abstract from the IoT Broker level, managing them for multiple organizations/tenants (which can be regarded as customers of Gateway services to manage several **IoT Devices**) via some API and/or Web user interface. Typical IoT Brokers are capable of managing only one organization. Thus, they are single-tenant in the sense that they broker messages using the topic concepts (which can be regarded as the key for subscription) without any internal partition of services as a sort of family of devices and subscriptions. Some IoT Brokers

---

[1] This chapter has been published as "Automating Heterogeneous Internet of Things Device Networks from Multiple Brokers with Multiple Data Models" in *Global IoT Summit.*

[2] *Acknowledgments:* The authors would like to thank the MIUR, the University of Florence, and the companies involved for co-founding the Herit-Data project. Km4City and Snap4City (https://www.snap4city.org) are open technologies and research of DISIT Lab. Sii-Mobility is grounded and has contributed to the Km4City open solutions.

can be multi-tenant, such as the FIWARE **Orion Broker**, which provides a partition of the devices in groups, and each of them may have a dedicated service/path for a specific scope (or of a specific customer). Furthermore, devices of different tenants could exist physically in different places (even having identical IDs), and the subscription to the broker's tenant may imply getting all messages/services in the partition. That is feasible only if the subscriber knows the identifier of the service path and, in the cases of access control, has the grant to access the services. The complexity of IoT Platforms grows with the need to manage multiple IoT Brokers, which can be managed by third parties different from the IoT Platform manager, i.e., **IoT External Brokers**, and/or directly managed by the IoT platform tools, i.e., **IoT Internal Brokers**, can be adopting different protocols, formats.

For the IoT External Brokers, the entities (IoT Devices) are directly registered on the IoT Broker, which is not under the control of the IoT Platform. Thus, the IoT Platform does not know the IoT Device data structure nor the composition of messages and services. Most of the IoT Platforms neglect these interoperability and integration aspects and provide simplified solutions. They do not care about internal/external brokers; they provide the possibility to set up end-to-end solutions with some restricted usage, for example, using only internal brokers they provide. Thus, AWS IoT by Amazon (AWS) [14] and Siemens MindSphere [101] make the broker structure transparent to their users unless they buy a specific add-on. While IoT Platform like Google IoT Cloud (Google IOT) [50] shows the Broker architecture but allows the usage of only one kind of protocol (e.g., MQTT). At least, solutions like MS Azure IoT (MS Azure) [77] or IBM Watson [60] are more flexible. MS Azure does not provide to cluster their objects, in other words, supporting only one organization on broker; the IBM solution does not allow the connection of External Brokers.

In summary, most of the solutions provide simple scenarios and mainly assume to have customers starting to use their solution from scratch (on cloud or on-premise), offering limited capabilities to integrate the platform with legacy IoT Broker and Networks deeply. Nevertheless, all of them can connect to other IoT Brokers and Networks using REST Call on API. In those scenarios, the third-party brokers are not directly connected and not managed regarding IoT Device registration, subscription, data storage, search, etc. Different IoT Devices connected to the same broker adopt the same protocol and may use various data models. If the message format is based on JSON, the corresponding schema may be defined and used for validation, while variables/attributes can be differently defined. For example, FIWARE Orion Broker adopts the NGSI protocol to manage the so-called **Smart Data Models** from which IoT Devices can be templated out [102]. The IoT Platform or Gateway must recognize and manage these **IoT Device Models** (registering, processing, producing, storage, etc.). In the case of IoT External Brokers, the IoT Platform may not know the IoT device models nor the identity (topic) of the External IoT Devices. Hence, at the arrival of a message from an unknown device (which can partially provide information in its body, typically not the metadata, since most of the devices minimize the data transmission), the IoT Platform is not in the condition to register the device, and neither to correct link the message to the former devices. Thus, the devices and messages are easily managed when a new external device is added to the IoT Platform if the data model adopted is known. In other words, if the IoT Platform knows the data model adopted by a device, it is easier to identify the data structure. So, it could automatically manage the relationships among data entities and verify coherence. For all these reasons, the Data Model concepts and formalisms are crucial.

In this chapter, the problems mentioned above and other aspects have been addressed in order to design and implement a solution for leveraging IoT network interoperability and the management of data models. To this end, Snap4City offers the IoT Directory concept and tool [28]. The IoT Directory supports (i) Internal and External brokers, (ii) the automated registration of devices managed into External Brokers' single- or multi-tenant services, (iii) automated registration by harvesting and reasoning of IoT Devices compliant with standard models such as FIWARE Smart Data Model, and any custom Data Model in Snap4City IoT Device Model providing a formal semantic definition of attributes.

The chapter is organized as follows. Section 3.2 presents the major requirements for the data model and broker interoperability in IoT Platforms. Section 3.3 shows the role of IoT Directory in IoT architecture for managing internal/external brokers with the aim of automated registration, management vs data models, and formal definition of their attributes. In Section 3.4, some details regarding the validation of the solution are reported. The validation included verifying the processing timing and giving a general numeric KPI about the shape of the entities.

3.2 Requirements analysis

This section lists and comments on the requirements an IoT Platform for IoT Network management and exploitation should satisfy. They have been identified in the context of workshops, conventions, and analysis for developing and using the Snap4City platform covering smart city and Industry 4.0 domains. The requirements for the IoT Platform are presented in *logical order* as follows. The following list of requirements also refers to a set of well-known platforms: AWS, Google IOT, MS Azure, Siemens MindSphere, and IBM Waston. Therefore, an IoT Platform should provide support to:

1. **Manage different kinds of IoT Brokers, IoT Devices, and IoT Edge Devices.** They should be based on different protocols, formats, and modalities to establish connections with the IoT Platform. Focusing on the Platform considered, they all support MQTT and HTTP, while Google IoT and Azure IoT support only MQTT Broker. It is important to highlight that most platforms provide specific components for different protocols; for instance, Amazon MQ supports Broker with AMPQP, MQTT, OpenWire, and STOMP protocols.

2. **Connect External and Internal Brokers.** They could be multi-service and could provide different protocols. Internal Brokers should be deployed and registered by the IoT Platform, while the External Brokers would be only registered to use them. In the Platform considered, the brokers are the products' core of stakeholders' offers; for this reason, the requirement is partially satisfied. In that sense, AWS IoT and Siemens MindSphere offer a paid add-on.

3. **Register, manage and use messages conformant to any Data Model with any data type.** Providing, receiving, managing, storing, and retrieving messages for any IoT Device of any Data Model with its attributes, data types, and related access control. A data model provides a model format for IoT device messages with several variables/parameters/attributes and their specific data types. For the listed Platforms, the messages from the IoT Devices are freely shaped to ensure data flexibility to the detriment of the data model. For example, Google IoT and IBM Watson use formats such as JSON or XML.

4. **Verify the correctness of IoT Messages of IoT Devices.** The

platform should be capable of verifying, including verification at the level of attributes, before accepting/sending them. Please note that each solution satisfies this requirement since the IoT Devices are formally defined at the registration phase.

5. **Semantic Interoperability.** This requirement is fundamental to achieving coherence among various IoT Devices (e.g., provided by different builders, addressing the same concepts and information on attributes). An IoT Platform should be capable of recognizing/classifying/retrieving information/attributes and behaving according to the semantic data model and types. For example, an IoT application should not risk misunderstanding the unit of measure assigned to attributes of different devices with the same name but different units.

6. **Support automatics deploy of Internal IoT Brokers.** The IoT Platform should support the automated deployment of IoT Broker internally managed. Thus, the platform directly manages internal brokers that register IoT devices. The result is an effortless experience for the user and an easy way to populate the network. This requirement can be implemented only if the Platform allows registering and managing new IoT Brokers. For this reason, this requirement is satisfied only in the Core of Siemens MindSphere and by all FIWARE Platforms for definition.

7. **Register External Brokers.** The platform must support the registration of IoT External Brokers. This means that the IoT Platform should be capable of registering IoT Devices/Services of the External Broker into the IoT Platform. Brokers can be single- or multi-tenant, and recovering the IoT Devices data model managed by the Broker is the first step in performing their registration. In the case of an external broker, the endpoint URL and the service and/or service path specifications would need to be subscribed to. None of the commercial platforms considered provides a solution for registering external brokers, thus allowing automated registration of their devices.

8. **Discover IoT Devices on IoT Brokers.** The platform must be capable of abstracting IoT Devices from their IoT Brokers and protocols. This is needed for their registration and classification, as well as search based on their position, nature, value types, units, etc. In other

words, it should be possible to discover/search (subscribe, get, send data) to/from IoT Devices independently from their position/connection in the IoT Network. The discovery process must be manageable in that its execution time can be scheduled, and it is possible with brokers that support a process for device discovery. The result should consist of an automated or semi-automated registration process for IoT Devices.

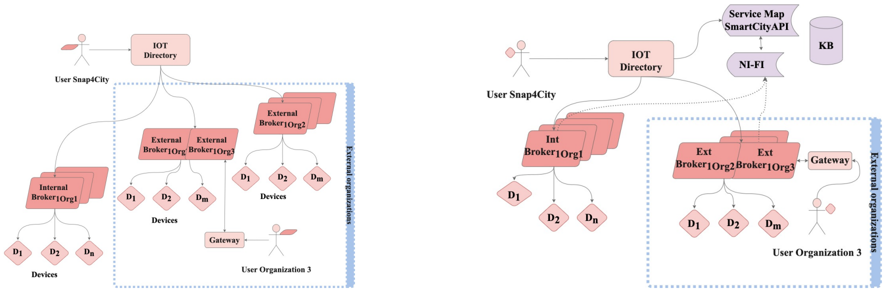
9. **Easy management graphic interface to list and test IoT Brokers and IoT Devices and query them.** For each IoT Device, it has to be possible to perform testing activities. As seen before, not all the Platforms mentioned above manage the IoT Broker unless they use a specific add-on. So, each is satisfied with this requirement only for the kind of devices they offer.

10. **Manage IoT Device Model and Device Data Type ownership and access grant.** This permits assigning/changing ownership and creating access grants to entities (Brokers, Devices, Models, etc.). In delegation management, it must be possible to list them (check the grants provided) and revoke the delegations. According to GDPR, any entity must start as private to the owner. The delegation should be possible for organizations, groups of users, and single users.

On other aspects, surveys about IoT Platforms are provided in [9,19,88].

3.3 The role of IoT directory in Snap4City architecture

The IoT Directory extends the features of generic IoT Platforms with the management of (i) IoT Data Models, (ii) IoT External Brokers, (iii) discovery of IoT Devices of External Brokers, (iv) support the multi-tenancy, (v) support several organizations, (vi) GDPR compliance, etc. In Fig.3.1a, the registration process of IoT Brokers (Internal or External) in the Snap4City Platform is reported, where the subscription in the Broker name denotes the organization. Several parameters are needed at the Broker registration into the IoT Directory, including the endpoint, security, name, External/Internal, single/multiple-tenant, etc. The most important difference for Internal/external Brokers consists of IoT Device management, as explained in the fol-



(a) a

(b) b

Figure 3.1: 3.1a registration of an IoT Broker, 3.1b registration of an IoT Device. The solid lines indicate the registrations, while the dashed lines indicate the data flow of the subscriptions

lowing. To be usable, the broker must be granted to each user or public [15]. A user only belongs to a single organization for security and privacy aspects.

Once a Broker is registered, the IoT Directory automatically subscribes the data platform to the new Broker for all its devices/topics so that each new message generated by the broker is directly brought to the data storage. In Fig. 3.1b, the subscriptions are denoted by dashed lines, while the registrations are shown as solid lines. In most IoT Platforms, the storage is called Data Shadow and allows the creation of historical data for IoT devices. In Snap4City, data storage feeding is performed by Apache NiFi so that the IoT Directory automatically associates NiFi with the topics of the new internal or external broker. This is due to the need for a robust, scalable tool with low latency to handle a huge volume of data entering the storage and coming from several devices and brokers.

In Fig. 3.1b, the processes of IoT Device registrations are depicted in both cases. In the case of Internal Brokers, the IoT Device registration is performed on the IoT Directory. The user may select the IoT Broker among those of the Organization and set several details. The registration may start with the exploitation of an IoT Device Model, the device ID, and then with the definition of GPS location, as well as all instance details. In Snap4City, the process can be performed: (i) on the IoT Directory via the user interface exploiting a model or not, (ii) exploiting API of the IoT Directory, (iii) using MicroServices in Node-RED, which are based on the same API of (ii), (iv)

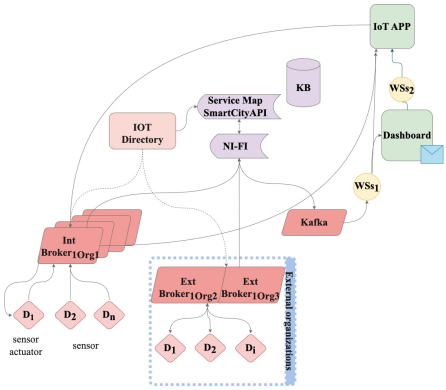


Figure 3.2: IoT Messages exchanged among entities: continuous lines are data flows, dashed lines indicate the tests that the user can perform to verify the IoT Devices/Brokers.

using a set of automated registration processes starting from Excel Files/tables. Each registered IoT Device is registered on the Knowledge Base, KB, (implemented with Virtuoso based on Km4City Ontology [20] for the semantic relationships and with Open Search for the time series data) with all its information and metadata (static information). The KB management allows to index the device and establish all the relationships with the other city entities located in the same area, place, city, region, road, GPS position, etc. This information would be very useful when new messages arrive from an IoT Device in the storage via NiFi (represented by the ServiceMap in Fig. 3.1b, with Smart City API for providing access to the storage). They are connected to the right IoT Device description and relationships. The correct and complete indexing and Smart City API are fundamental to enable the exploitation of IoT data by IoT Applications (Node-RED microservices [9]) and Dashboards [19]. Finally, Fig. 3.2 shows the data flow during the usage, illustrating the event-driven data flows. There are four ways to generate new IoT Messages from:

- **IoT Devices** pass from a Broker and are passed to (a) the NiFi, thus reaching the KB and storage, becoming part of historical data which can be accessed and queried from IoT App, Data Analytic and Dashboards; (b) Kafka to directly reach subscribed Dashboards via WebSockets, and IoT App.

- **IoT Apps** may be sent to IoT Broker or to Kafka front-end broker. Thus, the message can reach IoT Devices for acting on it, storage, IoT App, Dashboards, etc. If a sensor-actuator sends a message (Internal or External), his Broker broadcasts it to Ni-Fi, which spreads it in turn. The IoT Apps are also used for massive registration of IoT Devices and to perform data adaptation and ETL/ELT (Extract Transform/Load) processes.

- **Dashboards** is passed via Kafka toward the IoT App, to an Internal Broker, or direly into the storage. These messages can be regarded as virtual IoT devices that act on some sensors/actuators of IoT devices or even simulate them. The produced messages may be sent to internal/external brokers, and so on.

- **IoT Directory** may generate a new message towards an IoT Broker (and may also read the last message from the broker). The generation of messages from the IoT Directory is typically used to check if the Internal Broker is alive and works correctly and if the IoT Device messages are accepted.

  In the case of **registration of IoT Devices on an External Broker**, the IoT Directory does not manage the Broker, and thus, the Devices are registered in the External Broker by the third-party gateway manager without informing the IoT Directory. Therefore, the IoT Directory must recover the information required to register and index the devices in the KB. Then, the IoT Directory queries/harvests the external broker to get the structures of the IoT Devices (also called Discovery has to be started). The harvesting process may start with the broker API endpoint and the service path in the case of the multi-tenancy broker. The harvesting has to be performed at the broker registration and every time a new Device is added, thus a periodic Discovery is needed. For harvesting, the IoT Directory recognizes the data model and data types of the attributes and registers them in KB properly to validate them. The following subsections discuss the registration of IoT Devices on Internal and External Brokers. Please note that the registration of devices from External Brokers is one of the innovative aspects addressed by the IoT Directory, which can harvest the brokers and resolve semantic gaps on IoT device attributes/variables, see Section 3.3.2.

3.3.1 Registration of IoT Devices on Internal Brokers

The IOT Directory is fundamental for defining and registering IoT Device Models, i.e., data models. Focusing on the Internal IOT Brokers, the platform provides three ways to deploy IoT Devices:

- *Manual process:* the user can register an IoT Device using a graphic interface to input information. Registering a device based on a specific IoT Device Model and referring to a specific IoT Broker is possible.
- *Bulk process:* the user can upload a file with a list of Devices, defining the IoT Broker, Model, and Edge.
- *IoT App process:*The user can build an IoT APP using Node-RED on which specific nodes can be used. The nodes accept JSON with parameters related to the chosen Device Model to register new Devices.

The registered IoT Devices are shown in a table in which the users can manipulate only those they created, regardless of the generation process. The users can also see the public devices of the same organization in the list, while the general administrator has full visibility of all devices of all organizations.

3.3.2 Discovering & Registering IoT Devices from External Brokers

The Snap4City Platform allows the registration of External Brokers using the broker URL or the couple URL and service identifier in the case of multi-tenancy. After registering an External Broker, the user can start the harvesting process and choose the time period for the update. It is important to highlight that the Snap4City Platform may know a set of IoT Device Models (data models). Thus, in the best case, when the harvesting starts, it can recognize the device and its model (message format and device ID). **If the IoT Directory does not recognize the Device**, the Device has to be Registered. To this end the IoT Directory can query the Broker to have more information to register it. On the other hand, the IoT Device may or may not comply with a data model. If it is compliant, the registration is straightforward. It is not compliant; every single attribute has to be recognized in terms of Value Type, Value Unit, and Data Type (e.g., Temperature, Celsius, Float). The list of recognized/registered and non-recognized/non-registered Devices is presented. Through this interface, the user can resolve the problems manually by defining the missing data and enabling the registration.

The most common harvesting (automated registration process) problem is the lack of matching with known attributes. The IoT Platform tries to identify the attribute kind in terms of value type, value unit, and data type by performing query on data Dictionary and Km4City Ontology (via the connection from IoT Directory and ServiceMap by using Smart City API and Dictionary API. The recognition may have success in two cases: the data model is known but not this specific attribute or the model is not known, so all the attribute values of the Device are not recognized. In the first case, it is easy to fix the problem by applying a specific rule. In the second case, the Platform needs to learn a Rule for solving the attributes, thus defining a new Data Model in the IoT Directory. Formally, the processing rule R is defined in EBNF as follows:

```
R := "IF" <condition_list> "THEN" <action_list>

<condition_list> ::= <c> | <c> "AND" <condition_list>

<c> := <variable> <op> <constant>

<variable> := "device name" | "context broker" | "device type"
           | "model" | "value name"

<op> := "is equal" | "is not equal" | "is null" | "contains"

<constant> := integer | float | string | list

<action_list> := <a> | <a> "," <action_list>

<a> := <action_variable> ":" <action_constant>

<action_variable> := "Data type" | "Value type" | "Value unit" |
            "Editable" |<Coded Healthiness criteria> |
            <Healthiness value>

<action_constant> := string
```

The rule is divided into two parts: If statement and then statement. In the first part, the user can define the condition that describes a set of devices, e.g., a device name in common. The $< op >$ defines the operators; two of them ('is/ is not equal") can apply only on the number constant, and others ('Is null' or "contains") only on the string constant. In the second part, the user can establish the action that validates devices or values. In other words, for the subset of Devices identified by the $< conditionlist >$, the $< actionvariable >$ is modified as defined by $< actionconstant >$. An example of a Rule can be:

```
IF "context broker" is equal to "CONTEXTBROKER" AND
    "value name" is equal "activePower"

THEN "value type":"power",
    "value unit":"W", "data type":"float"
```

In this case, the rule's builder selects a subset of invalid attributes named active power, which has the Device subscripts a Broker named CONTEXBROKER, which allows the management of the multi-tenancy aspects. Then, it changes the value type, the value unit, and the data type in power, W, and float.

In the IoT Directory, searching and editing the saved rules is possible. When users save a rule, they must choose the broker to which it is applied. To cope with multiservice brokers, it is also possible to define a specific subset of services or service and service path. Thus, the application of a rule is associated with each specific Broker or Organization since a Rule can be suitable for an organization and not functional for others.

3.4 Validation experiments

Before proceeding with the validation test, clarifying the harvesting process and its associated cases is essential. Initially, the user must add the external broker to Snap4City. Upon doing so, the platform receives information about the devices. When harvesting begins, the devices share their models with the platform, which attempts to match the data model structure with the recorded ones. If a match is not found, the user has the option to program rules and can choose to save them. This allows users to filter data and apply multiple models to the devices. While this process may not be perfectly scalable, the same rule can be applied to several devices simultaneously. This approach aims to accurately index IoT devices, reduce registration time, and dynamically integrate newly registered IoT devices on the broker, thereby minimizing the gap between internal and external brokers. This is one of the primary objectives of the platform.

To validate this concept, we timed the harvesting process using an external broker with one tenant and several thousand devices (ranging from 30,000 to 50,000). We developed and applied new rules for managing the data models and repeated the experiment ten times to ensure a statistically significant sample.

The results indicate that users spend an average of 1.5 minutes filling out the form to add a new device with ten attributes, while the system takes approximately 3 seconds to register the device. When a user constructs a device from a model, the average time to complete the form decreases to under 1 minute, and the registration time drops to about 1.35 seconds. Additionally, when a user registers a new device through an IoT app, the system completes the registration in an average of 623 milliseconds.

Chapter 4

Malware Detection Using Federated Learning: An Investigation of Its Benefits

This chapter presents a novel approach to enhancing privacy and security in the Internet of Things (IoT) by utilizing machine learning techniques, specifically within the Federated Learning (FL) framework. FL is a machine-learning setting where multiple entities (clients) collaborate to solve a learning problem under the coordination of a central server (aggregator). However, emphasizing simplicity over robust security measures leaves IoT devices vulnerable to malware attacks.

Additionally, the balance between performance metrics (mean F1 score) and consistency (variance or standard deviation) is crucial for deployment decisions. This balance heavily depends on the application fields, especially given the rising use of adversarial training. For instance, high mean F1 scores in cybersecurity are often favored to enhance the detection of new threats, even if it results in performance variability. However, excessive inconsistency can jeopardize system reliability, making it vital to find a proper balance. Conversely, in medical diagnostics, consistency is typically prioritized over maximum performance to provide dependable, repeatable results under varying conditions, as inconsistencies might lead to serious misdiagnoses. Ultimately, the

decision to emphasize performance or consistency is influenced by the operational environment and the potential risks associated with mistakes, such as overlooked cybersecurity threats or inaccurate healthcare diagnoses.
The proposed solution involves implementing a federated machine-learning algorithm that utilizes a central aggregator and multiple clients. This approach, with its potential benefits, considers the computational capabilities of the devices to optimize learning without compromising data security, reducing data exchange and minimizing variance accuracy.
This study provides a thorough and comprehensive analysis using the IoT-23 dataset, which contains real and labeled examples of IoT malware infections. The desired outcomes are a reduction in performance variance and minimized overhead traffic. [1]

4.1 Introduction

The Internet of Things refers to a network of interconnected devices and objects that can collect, exchange, and analyze data, allowing them to communicate and interact seamlessly [84]. These devices feature lighter protocols, lower power consumption, and compact shapes, allowing flexibility and adaptability [12]. In addition, this kind of Internet can be implemented in several wireless networks in order to become adaptable in various use cases; for example, a LoRaWAN connection can be chosen in smart cities, while for smart homes, the Z-Wave strategy is more appropriate. Likewise, smart devices have a wide range of applications, many in sensitive areas. However, no standard supports all smart devices, and built-in security mechanisms are not yet standardized. Furthermore, there are no proven security methods to guarantee the digital security of these infrastructures [54]. In addition, IoT devices prioritize simple operation over robust security measures, making them vulnerable to malicious users who can coordinate attacks through malware designed to cause damage, disrupt, or gain unauthorized access to a system [39]. Therefore, it is necessary to implement an intelligent, light, and flexible solution that can learn various attack patterns and does not interfere

---

[1] This chapter has been published as “The intrinsic benefit of Federated Models in malware IoT detection” in *1st Workshop on Integrated Wireless Networking and Computing (IWNC).*

with the network's bandwidth or usability.

Furthermore, these methods must be able to forget learned patterns and re-learn when malware evolves [34]. In literature, deterministic techniques are commonly used in conjunction or not with Machine Learning (ML) methods, depending on the main aspect the tool should preserve. A key work that helps us understand this concept is offered by [83], which discusses anomaly detection analyses for different phases of the malware life cycle. Statistical analysis is utilized during the pre-execution phase, dynamic analysis is employed during the execution step, and memory analysis is conducted during the post-execution step. Conversely, malware detection is often considered a stand-alone problem and does not consider the network's topology or distributed nature. In [7], the authors take into account specific IoT challenges, such as privacy, but do not consider the impact of the model on the wireless network. Deeper, centralized models often are not appropriate for IoT cases because they can be accurate but may make the communication channel busy due to data exchange [46].

Federated learning is a suitable strategy to minimize data exchange. Furthermore, it is a distributed machine-learning algorithm that benefits from the topology of IoT networks. FL employs an iterative approach with discrete interactions between the client and server, known as federated learning rounds, to achieve results comparable to those of traditional machine learning models. Therefore, this method has the potential to access a large amount of data while maintaining privacy guarantees.

This chapter addresses the critical challenge of malware detection within the Internet of Things paradigm, focusing on the pivotal role of advanced models such as Federated Learning. I explore the adaptability of these models to the unique constraints of IoT environments and thoroughly examine the open challenges associated with deploying machine learning models in real-world scenarios where resources are finite and operational constraints are stringent. The approach centers on developing a federated machine learning classifier as a practical and deployable solution for identifying malicious traffic within IoT networks. Using data generated from traffic packet analyzers, I outline effective strategies for data definition, preprocessing, and the deployment of ML algorithms. This results in a robust model characterized by low variance in accuracy, making it suitable for resource-constrained environments. Additionally, I highlight the advantages of FL in reducing information leakage from devices and enhancing network performance. The findings underscore the importance of advanced ML techniques in strengthening IoT ecosystems against evolving cyber threats, ensuring the integrity of interconnected devices.

The paper is organized as follows: Section 4.2 defines the problem that forms the foundation for the proposed model and methodology discussed in Section 4.3. The main aspects and innovations of the proposed model are explored in detail. The 4.3.3section presents the results and their implications. Finally, I address future challenges and conclude the findings. The primary aspect and novelty of the proposed model are discussed. The following section discusses the results and their implications.

4.2 Problem definition: How can I minimize the variability in the results?

In the Internet of Things networks, achieving efficient malware detection while balancing key factors such as privacy, low computational overhead, and effective data exchange presents a complex challenge. The objective is to develop a robust malware detection method that provides high accuracy and maintains stability over time, with minimal variation in accuracy across different scenarios. This involves a state-of-the-art analysis to identify an optimal detection model and deploy it in a manner that balances the critical points mentioned above. The investigation analyzes a subset of overhead features from the IoT-23 dataset developed by Avast Software in Prague. The label distribution across the dataset is illustrated in Fig. 4.1, while a deep description is presented in Section 4.4. The challenges encountered included varying data set shapes, a significantly unbalanced label distribution, and a limited number of correlated features. A detailed data pre-processing investigation was also conducted. The objective of the research is to transform the limitations of IoT networks into strengths within a generalized model. Distributed approaches enhance scalability, whereas centralized models that exchange substantial data volumes between servers and clients can strain bandwidth. In federated models, only model parameters are shared with all clients, eliminating the necessity for each client to transmit their data and the relative parameters, thus reducing bandwidth usage and enhancing efficiency [35].

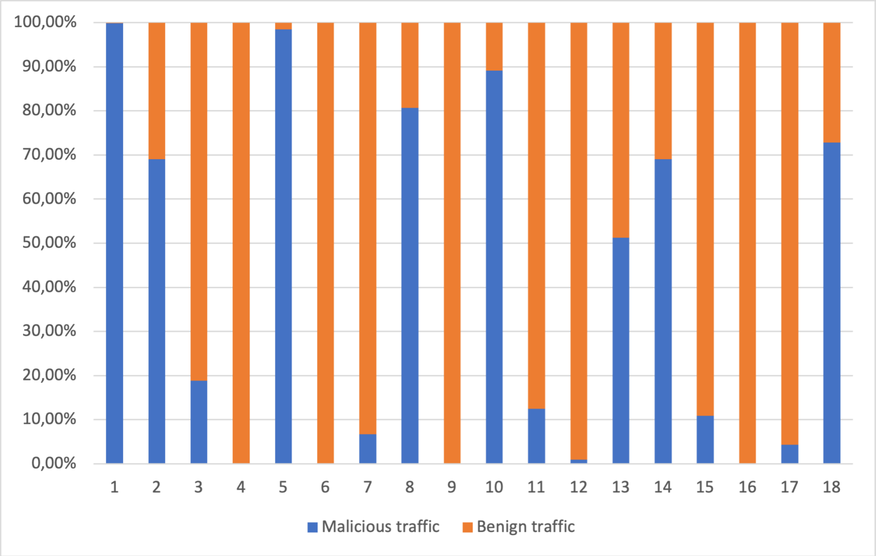


Figure 4.1: Distribution of Malicious and Benign across the clients (horizontal axis).

4.3 Methodology and solution proposed

In this section, I present the pre-processing data methods and models. I briefly explain the standardization methods and the federated models used in the classical and modified versions. In conclusion, the evaluation methods are presented to describe the analysis in detail.

4.3.1 Pre-processing data approach

The pre-processing methodology became crucial in this work due to the variety of data and the high correlation between the features. For this reason, I will introduce the core ideas that have been considered and their implications. The following paragraphs present local and global standardization and distributed principal component analysis. These techniques address the high variance present in the data and make them homogeneous to simplify classification.

Local and global standardization Standardization is a statistical technique that reshapes the data to make them more homogeneous and easy to classify. Given a client s with i examples in the dataset D_s, the local standardized data z_{si} is a transformation of the original data $d_{si} \in D_s$ as follows:

$$z_{si} = \frac{d_{si} - \mu_s}{\sigma_s} \tag{4.1}$$

where μ_s is the average vector of D_s and σ_s is the vector of standard deviation based on D_s. Otherwise, the global standardization replaces the μ_s and σ_s arrays with the μ and σ vectors, which are the mean and the standard deviation arrays calculated across all clients. These approaches offer valuable techniques for standardizing data, minimizing shared information, and enhancing model robustness.

The local standardization approach improves client confidentiality by modifying data within each client's domain. This method avoids transmitting sensitive information, such as the client's mean vector and variance, to the central server. Normalizing data locally helps address non-identically and independently distributed (non-IID) data across the network, making input data more uniform across clients. In contrast, global standardization involves estimating the overall data distribution on the server side. During the initial round of Federated Learning, the server aggregates mean and variance vectors from all participating clients, providing a comprehensive view of the network's data distribution. This global insight enables more robust model training by addressing disparities in data across clients. Both approaches offer distinct benefits: local standardization prioritizes privacy, while global standardization focuses on improving model performance by leveraging more comprehensive data distribution information. The choice between these methods depends on the federated learning application's specific needs and constraints.

Principal Component Analysis The distributed version of Principal Component Analysis (PCA) [81] is presented. A classical PCA aims to find a smaller dimensional subspace that captures as much of the variance of the data as possible. The method involves a collaborative approach where the variance matrix is constructed using data from multiple clients or nodes in a distributed system. In this framework, each client processes and summarizes its local data, which is then exchanged with a central server or coordinator responsible for aggregating this information into a global variance matrix.

Efficient computation of principal components necessitates the exchange of summarized indices or statistical measures between clients and the server. Nevertheless, this communication process may result in overhead due to data transmission and coordination. Despite the potential trade-off in increased communication complexity, the advantages of distributed PCA are consequential.

One advantage of PCA is its enhanced scalability to larger datasets that exceed the capacity of a single computing node or device. Distributing the computational load across multiple nodes reduces the overall processing time, leading to faster computation and analysis. Additionally, the distributed PCA model provides improved fault tolerance and resilience. Distributing data and computations across multiple nodes makes the system less susceptible to failures or disruptions in individual components. This redundancy can enhance the reliability and robustness of PCA computations in large-scale distributed environments. Furthermore, these distributed approaches also enable data analysis that preserves privacy. Clients can maintain control over their raw data while contributing only aggregated statistics or transformed representations to the central server. These strategies demonstrate proactive steps toward effective information management and collaborative model development within the context of FL. Each approach tackles unique challenges posed by distributed data settings, ultimately contributing to the advancement of secure and efficient machine learning practices.

4.3.2 Models

This section outlines the adopted models, introduces the base model, and describes my modifications. Specifically, the federated models discussed here are adaptations of Federated Averaging (FeAvg) and Federated Knowledge Distillation (FD). For a more detailed explanation of these models, please refer to [115], [94] and [62].

Classical version The federated models utilize an iterative approach that entails discrete interactions between clients, who possess a local model, and the server, which establishes the global model. The server chooses a subset of clients to train the local model in each iteration and updates the global model based on the parameters received. The logit function is the main distinguishing factor between the two approaches.

For the FedAvg algorithm is shown in Fig. 4.2, while the loss function of a generic s device is given by:

$$L_s(D_s, \mathbf{w}) = \frac{1}{N_s}\sum_{i=1}^{N_s} l(d_{si}; \mathbf{w}) \tag{4.2}$$

where D_s corresponds to the dataset of device s, N_s corresponds to the amount of examples list in D_s, l is an entropy function such as cross-entropy function and d_{si} is the i-th example of D_s. It is important to note that the weight $\boldsymbol{w}$ is usually a simple averaging of the selected clients' weights.

On the other hand, Federated Knowledge Distillation involves a more complex loss function and requires selected clients to store the mean logit vector per label during local training. A simplification of the algorithm is shown in Fig. 4.3. The logit is a combination of the previous logit and the distance between the client and server logit arrays, as defined by the formula:

$$L_{KD}(D_s, \mathbf{w}) = L_s(D_s, \mathbf{w}) + \lambda \cdot KL_{dist,y}(Y_{log}, Y_{log,s}) \tag{4.3}$$

where $L_s(D_s, \mathbf{w})$ was defined previously and $KL_{dist,y}(Y_{log}, Y_{log,s})$ is the Kullback-Leibler metrics distance used to compare the logit distribution of the client and the server. The pre-trained phase, typical of this method, was excluded from the experiment to create a lightweight and fast tool.

Modified version The difference from the usual methods is that the update of the local weights of device s at step $t + 1$ is calculated using the following formula:

$$\mathbf{w}_{s,t+1} = \beta \cdot \mathbf{w}_{s,t-1} + (1 - \beta) \cdot \mathbf{w}_t \tag{4.4}$$

where $\mathbf{w}_{s,t-1}$ is the weights vector of the device s at step $t-1$, and $\mathbf{w}_t$ is the average weights vector of the selected device at step t. β is a hyperparameter that can take values from 0 to 1. In the experiment, I used $\beta = 0.75$. This approach guarantees stable weights and robust model performance, enabling the model to learn effectively without being influenced by variations in scale, such as alternating between high-value and low-value clients across training rounds.

My methods assume that the system is stationary, meaning that the statistical properties of the data, such as the mean and variance, remain constant over time. In a stationary system, the data distribution does not

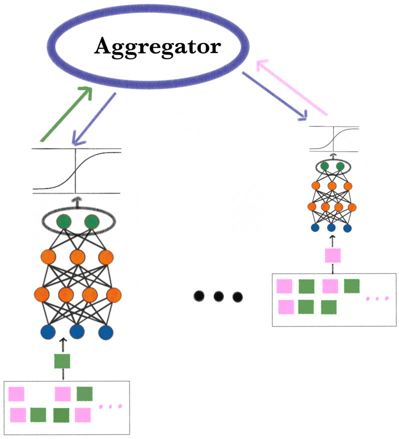


Figure 4.2: Representation of a Federated Average method. The blue arrows communicate the average weights vector, while the other arrows communicate the client weights vectors. The boxes indicate the devices.

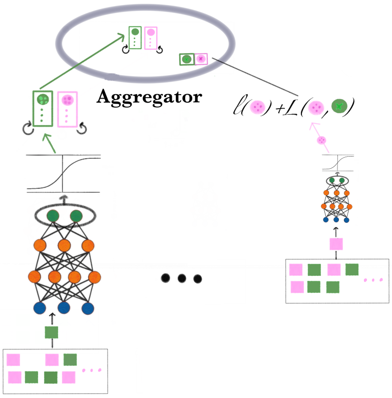


Figure 4.3: Representation of a Federated Distillation method. The circle represents the different logit vectors: the full-colored ones are the global logit, the circle with the x in the center is the contribution for the regularization, and the others are the client logits. The boxes indicate the devices.

change significantly, allowing the model to rely on consistent patterns. However, these estimates may become inaccurate if the global mean and variance shift over time. To address potential non-stationarity, I trigger an index update when there is a significant change in average loss. This mechanism helps maintain the accuracy of the estimates and allows for dynamic adjustments. Specifically, the update is initiated when the loss change exceeds a threshold value of 0.1 ($threshold > 0.1$). As previously highlighted, minimizing information exchange and efficiently preprocessing data are crucial for enabling rapid analysis and easy deployment in practical scenarios.

4.3.3 Evaluation approach

To facilitate the comparison of results, the area under the precision-recall curve (AUPRC) is chosen as an index of fitness. The AUPRC is a widely recognized machine learning metric for evaluating classification algorithms. Consider the unbalanced nature of the data and provide a suitable summary of the fitness of the model on the data [103]. This index helps to identify the balance between precision and recall at various thresholds. Precision is the ratio of correctly classified examples to all positively classified examples, while recall is the ratio of correctly classified positive examples to all positive examples.

When comparing different models, their AUPRC scores will be compared by ratio. For instance, let's consider a generic Federated Model (AUPRC(Fed)) and a Centralized one (AUPRC(Centr)), the ratio of their AUPRC scores are calculated as follows:

$$\mathrm{AUPRC}_{\mathrm{Ratio}} = \frac{\mathrm{AUPRC(Fed)}}{\mathrm{AUPRC(Centr)}} \tag{4.5}$$

A ratio greater than 1 indicates that the Federated Model outperforms the Centralized one regarding AUPRC, while a ratio less than 1 indicates the opposite.

In conclusion, a Chi-square test can be performed to verify the variance reduction and evaluate whether the variances of two populations are equal. By applying this test, I can establish whether the variance reduction is meaningful and quantify the effectiveness of the proposed methods. This evaluation offers robust metrics that assure improvement.

4.4 Dataset description

The selected data set is IoT-23, collected by Avast Software in Prague. This data set includes twenty-three captures, referred to as scenarios, of various IoT network traffic, gathered between 2018 and 2019.

In each malicious scenario, a specific malware sample was executed on a Raspberry Pi, utilizing protocols such as HTTP, DNS, DHCP, Telnet, SSL, and IRC. These scenarios were conducted in a controlled network environment, simulating conditions similar to those of an actual IoT device with unrestricted internet access.

The three most common malicious flow labels are as follows: PartOfAHorizontalPortScan (213,852,924 flows), Okiru (47,381,241 flows), and DDoS (19,538,713 flows). In contrast, the three least common malicious flow labels include C&C-Mirai (2 flows), PartOfAHorizontalPortScan-Attack (5 flows), and C&C-HeartBeat-FileDownload (11 flows).

For each capture, there are the original *.pcap* files and the corresponding Zeek *conn.log* file. This file is one of the outputs offered by Zeek software, a passive, open-source network traffic analyzer used as a network security monitor. It offers transaction data and extracted content data in the form of logs summarizing protocols and files seen traversing the wire. To manage the large traffic volumes each infection generates, .pcap files are rotated every 24 hours. Rapid file growth sometimes led to an early termination before the full 24 hours had elapsed. As a result, some captures vary in duration. Also, the dataset offers a *conn.log.labelled* file, that retains the flow of network connections but includes two additional columns for labeling.

These labels are one for the binary classes (Malicious, Benign), the other for the category of malware. In this work, only the binary labels are considered, and only the following features are considered:

- *duration*: How long the connection lasted
- *origin bytes*: The number of payload bytes sent by the originator. This is taken from sequence numbers for TCP and may be inaccurate (e.g., due to large connections).
- *missed bytes*: Indicates the number of bytes missed in content gaps, representing packet loss. A value other than zero will normally cause protocol analysis to fail but some analysis may have been completed before the packet loss.

- *originator packets*: Number of packets that the originator sent.
- *originator IP bytes*: Number of IP level bytes sent by the originator
- *responder packets*: Number of packets sent by the responder
- *responder IP bytes*: Number of IP level bytes sent by the responder (as seen on the wire, taken from the IP *total length* header field).

Of course, because of the nature of the experiment, dataset sizes vary significantly, with smaller sets comprising approximately four examples while larger sets contain over 100,000 samples. For this reason, the datasets are considered if they have at least 100 examples.

4.5 Evaluations

The experiments want to highlight how the federated models overcome the centralized model, which is used as a baseline. That happens because the proposed model is more suitable for the nature of the IoT environment and allows some key concepts. Also, the exchange data can be manipulated depending on bandwidth availability.

A summary of the results can be found in the box plots presented in Figure 4.4. These box plots illustrate the distributions of the AUPRC values for each data configuration and operation method. The PCA and standardized transformation results were aggregated for the centralized approach as they produced identical outcomes. In contrast, the federated approach without standardized data led to the presentation of both models' outcomes. Upon examination of the figure, it becomes evident that the federated model with globally standardized or PCA data exhibits minimal variance and demonstrates superior global AUPRC performance. A chi-square test is performed to verify this reduction in variance, and the results are reported in Table 4.1. The test results demonstrate that the Federated Learning approach significantly reduces variance in the Distributed PCA AUPR index and the Global Standardized AUPR index compared to the local Standardized Data Index, with a significance level of 0.05.

As mentioned above, analyzing model performances based on their AUPRC ratios provides valuable insight into different configurations. In the case of using only transformed data, FL methods exhibit significantly higher performance, approximately three times on average (with $\text{AUPRC}_{Ratio} \simeq 4.9$),

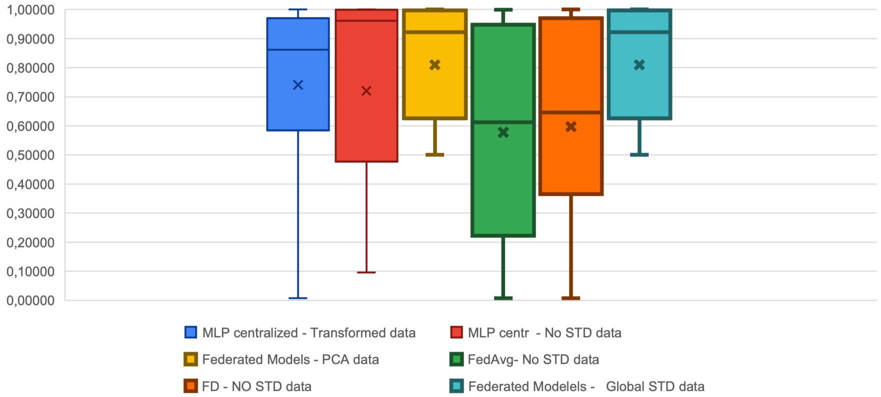


Figure 4.4: Box plots of the AUPRC of the clients. The cross in each box represents the mean. As we can see, the low variance is achieved by the Federated models.

Table 4.1: Summary of the average AUPRC ratio between Federated models (FedAvg and FD) and Centralized models

Model	Federated No STD	Federated Global STD	Federated PCA
Centralized No STD	0.94 (FedAvg) 1.07 (FD)	1.64	1.65
Centralized Data Transformation	0.82 (FedAvg) 0.97 (FD)	4.9	4.91

whether employing PCA-transformed data or standardized data. This outcome illustrates these methods' efficacy in reducing the variance across the results obtained. It can be seen that FedAvg demonstrates superior performance in this regard, as it minimizes the information exchange between nodes. In comparison to a centralized model, excluding standardized data during federated learning resulted in an average performance improvement of approximately 7% ($\text{AUPRC}_{Ratio} = 1.07$). Despite this improvement, the significant variance observed suggests that while this federated model shows promise, it may not offer the most consistent results.

Further comparison between standardized data and transformed data

Table 4.2: Result of Chi test on AUPRC index performed on the client AUPRC distribution

Chi test	p-value
PCA data	0.04
No STD data - FD	0.99
No STD data - FedAvg	0.99
Glob STD data	0.04

within the FL framework revealed a substantial performance gain, with an average improvement of around 60% compared to traditional centralized methods ($\mathrm{AUPRC}_{\mathrm{Ratio}} \simeq 1.6$). This highlights the critical importance of federated participation and data transformation in optimizing model outcomes.

A Chi-square test was conducted to validate these findings, revealing significant differences in the mean AUPRC distributions between standardized and transformed data, with p-values ranging from 0.1 to 0.15 (see Table 4.2). This indicates that, for PCA data and globally standardized data, the Federated model significantly outperforms the centered model in terms of both variance and mean. Additionally, data exchanged between nodes and the aggregator was analyzed. The results show that overhead traffic is directly influenced by the volume of data exchanged. Among the exchanged data types, model weights had the smallest size, averaging 224 bytes. In contrast, centralized models require the transmission of the entire device dataset, resulting in a significant overhead burden. However, the exact amount varies depending on device traffic and is not specified here. Other data transmitted during FL include feature sum and deviation vectors, each 56 bytes in size, which are essential for global standardization and principal component analysis (PCA). Basic traffic information is also communicated during the setup phase. Finally, the federated dropout (FD) algorithm requires additional data transfer, including the labeled logit vector (8 bytes) and model weights, further increasing the data exchange requirements.

The average data size for each client is approximately 140 MB. For centralized models, the average GPU utilization per example is 3.51 MB, while for Federated approaches, it is 2.15 MB. The average execution time for processing 1 MB is 5 seconds for centralized models and 4.83 seconds for

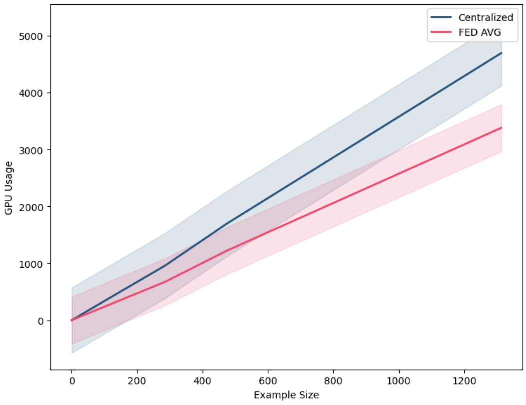


Figure 4.5: The graph shows the GPU usage (in byte) curve based on the test set size (number of examples). The blue curve represents the centralized model, while the pink curve represents the Federated model. The pink curve demonstrates the lower GPU usage of the Federated models compared to the centralized model.

Federated models. These values indicate that centralized models require greater computational effort for validation on individual devices. However, this approach allows for a more distributed workload across the network, as demonstrated previously.

Additionally, as shown in Figure 4.5, GPU usage for devices decreases with increased example size in Federated models compared to centralized models. This may also be attributed to the Federated model's strategy of using only a subset of data in each training epoch rather than the entire dataset.

Chapter 5

Traffic Data Images and Reduction of Training Sample Size

This chapter introduces classification attacks when the environment dictates strong constraints. The deployment of ML algorithms into the real world is complicated, and often, the network, the devices, or other stakeholders imply physical and reasoning limits, such as the impact on the network throughput.

Furthermore, the exponential growth of machine learning techniques in recent years has been driven by two key factors: increased data availability and the capability of automated optimization methods to improve precision. However, the construction of precise models often necessitates the use of extensive, heterogeneous datasets, which are not always readily available. The scarcity of data and the quality of training samples can profoundly impact a model's performance. Inadequate ML studies may lead to unreliable conclusions due to methodological weaknesses.

This chapter delves into two crucial ML techniques: adversarial training designed to bolster model robustness. Adversarial training fortifies a model's ability to handle alterations by subjecting it to intentionally modified input data that challenge the accuracy of its predictions.

5.1 Introduction

In the machine learning framework, the dataset is the starting point for all algorithms and is fundamental for the models' success. A significant subset of deployed models only works with specific datasets, so they do not generalize well to other datasets with similar data. Furthermore, factors such as data scarcity and the quality of training samples can influence the performance of classification models. However, it is essential to note that inadequate ML studies may lead to spurious conclusions due to a lack of familiarity and methodological rigor [124].

Various methods can be employed to enhance the robustness of learning models. One such method is adversarial training, which aims to improve a model's robustness and generalization by exposing it to intentionally crafted adversarial examples. These examples are slightly modified input data designed to mislead the model's predictions. By incorporating these adversarial examples into the training process alongside the original data, the model can better handle perturbations and deceptive inputs, ultimately improving its performance and ability to generalize to unseen data.

Moreover, adversarial training requires training on original and adversarial perturbed data to simulate real-world challenges. Ultimately, it emphasizes model robustness against specific threats.

This chapter investigates dataset construction and explores the relationship between data sample size and classification performance within network environments. It compares the effectiveness of synthetic image augmentation using Generative Adversarial Networks (GANs) and diffusion models in improving image recognition and classification accuracy.

Integrating traffic data with machine learning is a common practice to enhance system performance while preserving privacy. However, ensuring the reliability of machine learning systems relies heavily on data quality. Discrepancies between training and evaluation datasets can lead to unreliable and unsafe outcomes, especially in safety-critical applications. To address this concern, I propose introducing a 'conformance characteristic' during the data management stage. This characteristic ensures that the data used for training and evaluation conform to the same standards, promoting consistency and reliability in the model's performance.

I use biological methods that leverage principles similar to population biology to determine the minimum sample size required for robust model performance. These methods involve studying how to handle data scarcity when the distribution within a dataset is unknown. We can define a suitable sample size for practical model training and generalization by assuming distribution over the good of fitness indices.

In summary, this study underscores the crucial role of robust dataset management in maintaining the integrity and effectiveness of machine learning applications in networked environments. Advanced techniques, including GAN-based synthetic data generation and biological principles, are employed to optimize dataset size for optimal performance. The significance of expertise in dataset management is critical for the success of machine learning applications, as the knowledge and experience in this area are highly valued in the field [27].

Moreover, the proposed method of rapidly gathering and classifying data over an extended period presents notable advantages for balancing data availability and classification accuracy, particularly in IoT malware detection. Shortening the data collection phase minimizes resource consumption - such as storage, bandwidth, and processing power - which are often constrained in IoT settings. This efficiency is pivotal for sustaining system performance and reducing operational costs while still ensuring sufficient data capture for meaningful analysis.

Furthermore, the extended classification period enables the system to leverage temporal patterns and behavioral trends that may remain undetected during shorter observation intervals. This prolonged analysis enhances the detection of intricate and evolving malware threats by providing a broader context to discern subtle, persistent, or stealthy attack patterns. Ultimately, this method optimizes the interplay between data collection and classification accuracy, ensuring that reduced data input does not compromise performance. Instead, it emphasizes strategic feature selection and aggregation to derive valuable insights from smaller datasets, thus facilitating high detection accuracy without overwhelming infrastructure in resource-limited IoT environments.

This chapter follows a standard structure with several key sections. The first section presents the state-of-the-art in the field, providing an overview of current research and advancements. The subsequent section, section 5.2, details the problem definition and experimental setup, outlining the research questions and methodologies used. Section 5.3 presents the results obtained from the experiments conducted, accompanied by insightful comments and interpretations of the findings.

5.2 Methods and experiments

This section presents the core idea and experimental setup. The following subsection outlines the procedure for determining the sample size for each class. This determination is based on the f-score, which measures the model's discriminatory power; the level of confidence, which indicates the likelihood of the results being within a specific range; and the sample size effect, which is the magnitude of the difference between classes. The other subsection describes the methods used to generate the dataset, the crucial process for the adversarial methodology.

5.2.1 Sample size problem definition

As previously stated, the definition of sample size is crucial in many fields, especially in biological research, where collecting data is expensive and complex. Consequently, the literature offers a variety of solutions, encompassing both statistical and machine learning methodologies. This section will illustrate the two sample size definition methods. These methods differ in their initial assumptions. The first method assumes a distribution over the indices, so it is a parametric approach. In contrast, the second is an innovative non-parametric approach, a novel concept in our field based on the empirical distribution of correctly classified examples.

Delving into the practical implications, the authors of [41] examined the impact of the size of the test set on the accuracy assessment and comparison in classification evaluations. Their approach, which assumes a normal distribution on a specific good of fitness index to define its confidence interval and obtains the sample size by inverting that formula, has direct applications in real-world scenarios. With this approach, the only parameter to choose is the confidence level of the interval. In the formula we have:

$$h = \sqrt{\frac{z^2_{\alpha/2} \cdot M(1-M)}{n}} \Rightarrow n = \frac{z^2_{\alpha/2} \cdot M(1-M)}{h^2} \tag{5.1}$$

M indicates the value of the performance metric, h represents the desired confidence interval half-width. The critical value of the normal distribution for the two-tailed significance level, α, is represented by $z_{\alpha/2}$.

However, the distribution assumption implies linearity, which is not empirically demonstrated to be true. For example, the index curve during the training phase has no linear trend. This non-linear trend suggests the need for a more flexible distribution, such as a Beta, as shown in [22]. This is a key consideration in our method.

Our method does not assume a distribution on the index to overcome this issue. The new proposed method is based on the confusion matrix, the most common tool of a classification model evaluation. Hence, all performance indices are calculated on it. The underlying concept is to consider the confusion matrix as the *empirical joint distribution of the model's performance.* Each matrix's no-diagonal element represents the probability of belonging to a class i and being assigned to a class j. In contrast, the probability $P(TrueClass_i \cap AssiClass_j)$ measures the model's ability to discriminate when $i = j$ (i.e., on the diagonal). In addition, we can write the good of fitness index, F_ϕ score, depending on the true positive and the false positive rate. The unbalanced label distribution motivates the choice, and this index's robustness respects outliers. In addition, it is calculated for each class in case of a multivariate problem. Considering the formulation of the index, we have:

$$F_\phi = \frac{(1+\phi^2)\cdot \text{true positive}}{(1+\phi^2)\cdot \text{true positive} + \phi^2 \cdot \text{false negative} + \text{false positive}}\,. \tag{5.2}$$

The terms type I error (α) and type II error (β) are often used interchangeably with the general notion of false positives and false negatives. Therefore, we can define the F-score for each class and determine their sample size using the above definition. An appropriate solution to address the goals is a non-parametric McNemar-Bowker test, with matched pairs of subjects, to determine whether the marginal frequencies are equal, i.e., whether there is *marginal homogeneity*, in the formula:

$$P(TrueClass = Class_i) = P(AssClass = Class_i) = \frac{n_i}{N} \tag{5.3}$$

Under the null hypothesis with a sufficiently large number of outliers, the static test is distributed according to a chi-square distribution with one degree of freedom. In order to perform this test and thus obtain the sample size of each class, it is necessary to know Cohen's effect size (d). This index makes the association between the features explicit and gives an idea of the data set's composition.

Once the requisite sample size has been defined, we will generate the dataset, which a deep Convolutional GAN and diffusion methods perform.

Why McNemar-Bowker test? The selection of the nonparametric McNemar-Bowker test for analyzing the confusion matrix in malware classification brings several benefits, especially in tackling the issues associated with imbalanced datasets. One major advantage of this test is its nonparametric characteristic, which means it does not rely on any assumptions regarding the underlying data distribution. This feature is particularly advantageous in malware classification, where data distributions can vary significantly and unpredictably due to the continuously evolving malware threats and the variety of IoT settings. The null hypothesis of the test assesses whether the marginal distributions of the confusion matrix are identical, which, in classification terms, means evaluating whether the actual and predicted label distributions are statistically alike. This enables a thorough evaluation of classifier efficacy by pinpointing any systematic biases in the classification. Conducting the McNemar-Bowker test separately for each class and considering its implications on Type I errors offers a versatile method for addressing the class imbalance. In datasets with significant imbalance, where some malware types may be less represented, this method permits strategic decisions, sometimes allowing minor imbalances to prioritize overall model performance. Furthermore, the test can aid in data augmentation efforts by providing insights into differences in class distributions, which can be utilized to rebalance the dataset effectively. In summary, the McNemar-Bowker test delivers a strong statistical framework for assessing classifier performance without being hindered by parametric assumptions, thus serving as an essential resource in situations involving imbalanced datasets and complex classification challenges in IoT malware detection.

5.2.2 Image generation

The dataset considered is EDGE-IIOTSET, which is the same as the previous chapter and collects traffic attack data. In this analysis, the continuous raw traffic is collected in pcap files, which are then split into multiple discrete traffic units and converted into images. The data granularity is the flow, defined as all packets with the same 5-tuple, i.e., source IP, source port, destination IP, destination port, and transport-level protocol. The [118] tool plays a crucial role in this process. It takes the pcap files as input and performs traffic anonymization/sanitization, which randomizes the MAC address and IP address in the data link layer and IP layer, respectively. It then removes duplicates and traffic without the application layer. Once the data

Table 5.1: Class Distribution of test set. It is important to note that the test set contains more neutral traffic examples than Malware traffic.

Class	Count
DDoS Flooding Attack (ICMP)	60000
Neutral Network	291510

are unique and anonymized, only the header is considered, and all flow pcaps are concatenated until they are large 743 bytes. This is because the most relevant information is concentrated there as demonstrated in reference [118]. However, if the file size is shorter, the 0x00 byte is appended to the end to achieve the desired length. The resulting files of the same size are then converted to grayscale images. In these, each byte of the original file represents a pixel, with 0x00 representing black and 0xff representing white. Each grayscale image is 784 bytes, with dimensions of 28 by 28.

For this study, we focus on neutral data and data related to a DDoS flood attack using the IMCP protocol. The distribution of the real image dataset is shown in Table 5.1.

Adversarial learning, a pivotal collection of models and methodologies, fortifies neural networks' robustness regarding security. Unlike conventional machine learning techniques that assume the training and test data are generated from the same statistical distribution (IID), adversarial learning acknowledges the real-world scenario where users may intentionally supply simulated data. By learning from these adversarial examples, the neural network can be trained to be robust against such attacks, ensuring its reliability in high-stakes applications.

For the synthetic data, a deep convolutional generative adversarial network (DcGAN) is performed for each class because the fake neutral traffic tests the adversarial training. The methods are applied to both cases for a clear and deep comparison. At the core of this network are two principal components: a generator tasked with creating new images and a discriminator responsible for classifying images as authentic or fake. In essence, this network is designed to solve a min-max optimization problem, with the objective being to maximize the loss of the discriminator and minimize the loss of the generator.

Diffusion models introduce a novel approach to image generation, where random noise is progressively refined into a realistic image. The training process involves a neural network that iteratively learns to *de-noise* a data set of real images, thereby teaching the network the statistical properties of natural images. During generation, the model starts with a noise image and progressively removes noise through a series of steps, introducing structure and detail until a realistic image is achieved. This method offers a potent tool for generating high-fidelity images. It has the potential to be applied to a variety of image manipulation tasks, sparking new possibilities in the field.

In conclusion, the models meticulously generated 234 images for each class, demonstrating high precision and control over the construction of the other datasets. This meticulous approach provides a solid foundation for the research, instilling confidence in the results and their potential applications.

5.3 Experiments and results

This section outlines the experiments, starting with determining sample size and concluding with comparing the results.

The methodologies discussed can be used to establish the sample size for each class. In the case of the Foody approach, the sample size is calculated as the inverse of a confidence interval. To apply the formula, the parameter values need to be fixed. For instance, with an F1 score index of 0.9, $z_{\alpha/2} = 1.95$ (where $\alpha = 0.95$), and a span interval of 0.05, the formula is defined as follows:

$$n = \frac{1.95 \cdot 0.9(0.1)}{(0.05)^2} = 139 \tag{5.4}$$

In contrast, with my new methodology, the needed steps are fixed the $\phi = 1$, $F_1 = 0.9$, and the $\alpha = 0.05$, so we have:

$$\begin{aligned} \beta &= \frac{(1+\phi^2) - F_\phi(1+\phi^2+\alpha)}{1+\phi^2+F_\phi} \\ \beta &= 0.0724 \end{aligned} \tag{5.5}$$

In the end, applying the McNemar-Bowker test requires an effect size estimate. In my analysis, this estimate is based on the Foody sample size for simplicity. Hence, once the total sample size has been estimated using the Foody formula, it can be used to train a simple model for a few epochs (10 epochs). The sample size outcome is 45 examples for the class.

Table 5.2: Sample size estimates.

Method	Sample Size
Normal assumption	139
F1 for balance case ($\hat{d} = 0.482$)	90

A base model with fixed parameters and hyperparameters was selected to investigate the effect of sample size and the impact of synthetic data on model performance. In particular, a simple convolutional neural network was considered. The model was trained for 100 epochs, after which the F1 score was calculated on different test sets. In order to facilitate a clear comparison, three distinct training sets are employed: the dataset defined by Foody's formula, the dataset based on our approach, and the entirety of the original dataset, which is utilized for testing purposes. Furthermore, the training, as mentioned above, sets can be divided into two distinct groups based on the estimation methods employed. The sample sizes associated with each group are presented in Table 5.2. Each dataset mentioned above has several versions, each with a distinct percentage of fake images. For the data augmentation trainsets, we considered four distinct cases: 0%, 20%, 30%, and 50%. Concerning adversarial training, malicious images were replaced with fake benign ones. To ensure a comprehensive investigation, we conducted 50 repetitions of each experiment. This approach enabled us to calculate the mean and variance of the performance metrics. Additionally, we randomly sampled the original dataset for each trial to create the dataset under consideration.

5.3.1 Results

In this section, the results are presented and commented on. In particular, the outcomes of our experiments are shown in Table 5.3. This table comprehensively evaluates adversarial training utilizing different techniques (Diffusion Method and GAN) across the datasets presented in the above section. The primary metrics used for comparison are the average F1 score across the classes, which measures the balance between precision and recall in the context of adversarial training, and the variance, which indicates the consistency or stability of these methodologies in achieving performance.

The initial approach, Adversarial Training, was implemented using the

Table 5.3: The results of the experiments are presented in the following table. The values represent the average F1 score on 20 trials, while the values in the breaks represent the variance.

Methodology		**Mean of f1 score**	
		Foody	**Proposed**
Adversial Training	**Diffusion method**	0.637 (0.058)	0.586 (0.048)
	GAN	0.453 (0.077)	0.453 (0.050)

Diffusion Method. The dataset based on Foody simple size demonstrated a mean F1 score of 0.637 with a variance of 0.058. In contrast, the proposed method (referring to the new approach or technique introduced for comparison) yielded a slightly lower mean F1 score of 0.586. Still, it demonstrated improved consistency with a variance reduced to 0.048. This indicates that while the proposed method may exhibit a slightly lower average performance, it proves more stable and predictable results than the baseline Foody dataset.

Similarly, implementing Adversarial Training with GAN on the dataset derived from Foody's theory yielded a mean F1 score of 0.453. This result was maintained with the new method, although with a reduced standard deviation of 0.050. This underscores the potential of the proposed approach to achieve comparable results with enhanced stability, paving the way for its promising future applications.

In conclusion, it can be seen that there are many different ways of achieving good results, depending on the methodology and dataset being used and how the results are defined. This underscores the importance of choosing a methodology that suits the specific needs of the task and dataset being used, empowering researchers and practitioners in their decision-making. The proposed methods have shown some encouraging results regarding stability and reliability, mainly when using data augmentation techniques with GANs.

Chapter 6

Denoising Diffusion Models for Improved Malware Detection

In this chapter, I continued the investigation of malware classification in a poor information environment. Existing malware detection strategies often concentrate on specific facets, such as efficient data collection, particular types of malware, or handling data scarcity. While valid, these strategies typically overlook the potential for minimizing sample size, focusing instead on data augmentation. This work uses the novel method to determine the minimum sample size necessary to achieve a specified accuracy level, measured by the F1 score derived from the confusion matrix. The focus is on TCP header traffic data transformed into images through flow-splitting techniques for multi-class traffic classification. In addition, the diffusion model are performed to generate new synthetic traffic images and show that my method outperforms existing techniques in terms of stability and predictability. This study also compares the effectiveness of synthetic image augmentation using Generative Adversarial Networks (GANs) and Denoising Diffusion Probabilistic Models (DDPM) in improving image recognition and classification accuracy. [1] [2]

---

[1] This chapter has been published as "Addressing Data Security in IoT: Minimum Sample Size and Denoising Diffusion Models for Improved Malware Detection" in *CNSM2024*.

[2] *Acknowledgments:* This work has been partially supported by the National Science Foundation (NSF) awards # 2133407 and # 2201536.

6.1 Introduction

Securing Internet of Things applications is important but challenging for several reasons. One of them is the inherent limitations of low-power devices, which cannot adequately address robust protection strategies. Furthermore, consider the scenario in which an attack persists due to a lack of immediate identification: in such instances, the system may be severely compromised, potentially leading to significant financial losses for the network owner, as highlighted by [55]. This is particularly problematic because it compromises sensitive data and threatens physical and network security.

The application of machine learning has become increasingly prevalent in the detection of malware and other cyber threats [104], given its ability to process large data sets and decipher complex relationships between system variables [95]. Several ML approaches have been proposed to effectively detect previously seen and unseen malware, securing (IoT) networks [97].

While these approaches have merit, an accurate model often requires the availability of extensive and heterogeneous datasets, that are not always available or accessible. In addition, factors such as data scarcity and quality of training samples can affect the performance of classification models. As highlighted in [73, 121], the spurious correlation due to a lack of data and methodological rigor may lead to erroneous conclusions. Even when the ML model is rigorous, having a correct training dataset is the key.

To address a robust model and limit spurious correlation, the literature investigates methods to determine the minimum sample size [73, 121]. In addition, in network security literature, a common solution in case of data scarcity is to implement a data augmentation strategy that enhances the learning model's robustness. Data augmentation involves artificially expanding a training dataset by generating modified versions of the original data without requiring new data collection [99]. Introducing noise and data variability, these techniques enhance the model's generalization ability and reduce overfitting [113]. A strategy to apply this technique is to feed the model with traffic transformed into images. In [79], for example, feature extraction from based-traffic-images is shown to be not only more efficient in terms of accuracy compared to the binary representation of the same packet but also allows the method scalability. As demonstrated e.g., in [82], a low-dimensional representation of the image simplifies its categorization. Motivated by these results, the work shows how new image generation techniques, such as diffusion model-based [27], combined with dataset management, can achieve higher accuracy and lower false positive rates in malware classification, compared to existing approaches. The core idea of my algorithm involves collecting data in short intervals and classifying traffic over larger temporal windows. This approach emphasizes the importance of determining the optimal sample size when utilizing data augmentation. In particular, I performed the previously mentioned approach to find a minimum sample size based only on the confusion matrix to address the lack of sufficient training. Moreover, I adopted Denoising Diffusion Probabilistic Models (DDPM) for data augmentation and dissected its link with the false positive rate. *I found that the DDPM-generated data have a 7% higher F1 score and less variance (5%) than the Generative Adversarial Networks (GAN) generated data* and a higher average AUROC index along the classes. Also, through explainable AI techniques, I show how DDPM images are more similar to real with respect to those generated with GAN, hence validating our approach.

The rest of the chapter is organized as follows. Section 6.2 highlights the problem definition and our contributions. In Section 6.3, I describe the experimental setup, detailing our methodology. Section 6.4 discusses the experimental results, providing insightful comments and interpretations.

6.2 Problem Definition: Addressing Data Scarcity in IoT Malware Detection

In traditional approaches (for IoT attack classification with or without complete or sufficient data), the accuracy of a model is directly linked to the amount of data it is exposed to. Researchers have shown that more data points allow the model to recognize a wider range of patterns and establish a more resilient understanding of the underlying relationships [122]. However, in the context of IoT malware detection, data is frequently lacking and costly to obtain, making it difficult to achieve accurate classification. I aim to address this challenge by collecting data over a short period and then classifying it over a longer timeframe. The objective is to categorize different types of malware when data collection is limited or expensive. To identify the minimum amount of data to collect, I employed my method based on the confusion matrix, without distribution assumption, to obtain accurate results regarding model accuracy and false/true positive rates. Data augmentation uses the diffusion probabilistic model to create more stable, high-fidelity images.

6.3 Methodology and Solution Design

Data scarcity, or the availability of only a few datasets, is a common issue in malware classification. To address these, I perform my method based on the confusion matrix and F-score. Our process does not need an assumption on the data distribution and identifies the minimum number of experiments for each class thanks to the McNemar-Bowker statistics test [38]. Furthermore, according to the data scarcity hypothesis, I explore the Denoising Diffusion Probabilistic Model to increase the data variability and make the classification model robust and precise [57].

6.3.1 Minimum Sample Size Definition

The literature provides some examples that explicitly explain the relationship between the number of examples considered for the ML analysis (sample size) and the performance of the model [10]. For example, models with fewer parameters and a more straightforward structure may perform well on smaller data sets, as highlighted in [51].

As shown before, my method does not assume a distribution on the index. It is based on the confusion matrix (CF) since all performance indices, such as the F1-score, are calculated on it. The underlying concept is to consider the CF as the *empirical probability joint distribution function of the model's performance* and perform a statistic test considering a given quantification of accuracy through the definition of test errors. Note that each CF element, except the diagonal ones, represents the probability of a mismatching, e.g., belonging to a specific malware such as a Distributed Denial of Service (DDoS) and being assigned by classification model to another malware class e.g., a Port Scanning attack. In contrast, the diagonal element measures the model's ability to discriminate well. The more suitable test for our scope is the nonparametric McNemar-Bowker test [38].

Focusing on our specific case, given the imbalanced nature of the malware dataset, the F1 score is an appropriate choice of performance index. Concerning the generic F1 score, it can be expressed in terms of the values of true positives (TP), false negatives (FN) and false positives (FP), as shown in the following equation:

$$F_1 \geq \frac{2 \cdot \mathrm{TP}}{2 \cdot \mathrm{TP} + 1 \cdot \mathrm{FN} + \mathrm{FP}}\,. \tag{6.1}$$

Thanks to the above approach, we can define the F1 score for each class and determine their sample size by applying the above definition to the test. Since a type I error (α) is a false positive conclusion in a statistic test and a Type II (β) error is a false negative conclusion, these terms are often used interchangeably with the general notion of false positives and false negatives mentioned in the formula.

6.3.2 Flow Image Generation

This subsection provides a concise overview of the generative models employed. In particular, the Deep Convolutional Generative Adversarial Network (DcGAN) and the denoising diffusion probabilistic model are considered. In the literature, the GAN model is the most common solution for data augmentation of traffic images. This approach is generally based on two distinct components: a generator generating new images and a discriminator classifying real and fake images. In our case, the generator learns how to build the traffic image from the original, while the discriminator validates the similarity between artificial and real traffic-based images. The network is designed to solve a min-max optimization problem. The objective is to maximize the loss of the discriminator (assuring the similarity between real and fake images) and minimize the loss of the generator, which implies a network that knows how to mimic the traffic behavior.

In our analysis, both losses are Binary Cross Entropy. Several significant issues afflict a significant number of GAN models. Firstly, non-convergence occurs when the model parameters oscillate, destabilize, and fail to converge when the losses of the model are in accordance. Secondly, model collapse happens when the generator produces a limited variety of samples, effectively failing the diversity of the generated data. Another issue that may arise is the diminished gradient. This is present when the discriminator becomes overly successful, causing the generator's gradient to vanish, leading to a lack of learning. Finally, GAN models are highly sensitive to hyperparameter selections, which makes them challenging to optimize effectively [98]. The DcGAN is applied in this work due to the computational benefits and their ability to capture semantic features, as demonstrated in [49]. The distinctive

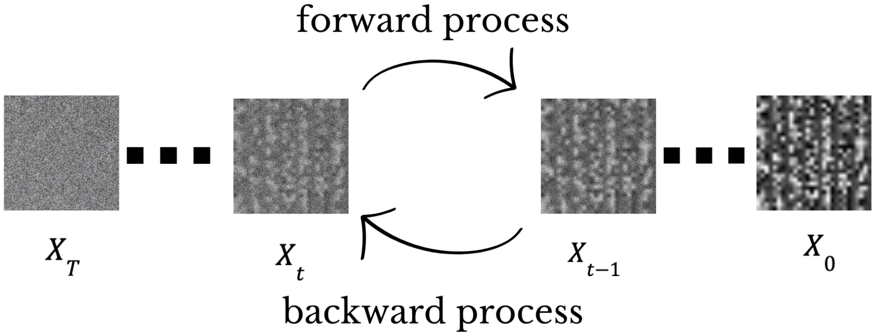


Figure 6.1: The Denoising Diffusion Probabilistic Models' mechanism involves adding noise in the forward phase and learning noise removal techniques in the backward phase, without assuming data distribution

feature of the DcGAN is the generator's use of a convolutional network. In our case, the convolutional functions are one-dimensional, reflecting the data's black-and-white nature and simplicity, ensuring a minimum cost in calculus.

In recent years, the diffusion model has emerged as a competitor to the GAN model [32]. The novel approach to image generation entails the progressive refinement of random noise into a realistic image. This model is a parameterized Markov chain trained through variational inference to generate samples that closely resemble the original data after a finite number of steps, as described in [56]. The model learns the transitions of this chain, which is a Markov chain that incrementally adds small amounts of Gaussian noise to the data in the opposite direction of sampling (forward process) until the original signal is destroyed (backward process). Hence, it is sufficient to set the transitions of the sampling chain to a Gaussian distribution conditioned to the images used as input. This process is known as diffusion and is illustrated in Figure 6.1. Moreover, the ability to master the backward process enables the construction of a model capable of generating images from pure random noise. Consequently, this approach allows for a particularly simple neural network (NN) parameterization by θ.

The network takes two inputs, X_t (the final traffic-based image) and t (the steps that we want), and outputs a vector $\mu_\theta(X_t, t)$ and a fixed matrix $\Sigma_\theta(X_t, t)$, respectively the mean of the pixel and their variance matrix. This allows each step in the forward diffusion process to be approximately reversed by $X_{t-1} \sim N(\mu_\theta(X_t, t), \Sigma_\theta(X_t, t))$. This method offers several advantages and addresses all the DcGAN issues listed above, including the ability to easily tune the model and generate more stable and performing outcomes in terms of image fidelity. In addition, the learning model to implement this kind of model can be chosen appropriately for the task.

In literature, several NNs can be chosen, but in our case, the most suitable choices are simple and lightweight ones that can capture the key aspect of traffic-based images and give an at the same time. For this reason, this work considers a simple U-Net [100] with a 1-dimensional convolution layer. This choice is due to its ability to retain spatial information, which aids in the recovery of fine details and precise object localization. By adopting a U-shaped architecture, U-Net efficiently combines local and global information, improving semantic accuracy concerning Convolutional Networks and the most common NN, such as YOLO, which is slower and heavier. Moreover, the U-Net reduces the number of parameters compared to the above-mentioned networks, allowing faster training convergence and requiring fewer computational resources. The capacity of the U-Net architecture to learn effectively from limited data and to handle images of varying sizes without preprocessing further contributes to its versatility and practicality [100].

6.4 Evaluation

The following sections explain how the application of a new method to define the sample size combined with data augmentation through the adoption of DDPM can achieve high results in terms of F1 score, taking into account the False positive rate. Also, a deep study of synthetic images using explainable AI techniques demonstrates how DDPM helps the classification model.

6.4.1 Experimental Setting

Figure 6.2 shows a summary of the experiment workflow. The packet capture (PCAP) files collected in *EDGE-IIOTSET* [40] and *Malicious Network Traffic PCAPs and binary visualization images Dataset* (MNT) [26] are considered and analyzed as images for this work. Specifically, the files are initially divided into unidirectional flows based on the same 5-tuple: source IP, source port, destination IP, destination port, and transport-level protocol. Subsequently, the TCP headers of each flow are concatenated. Addition-

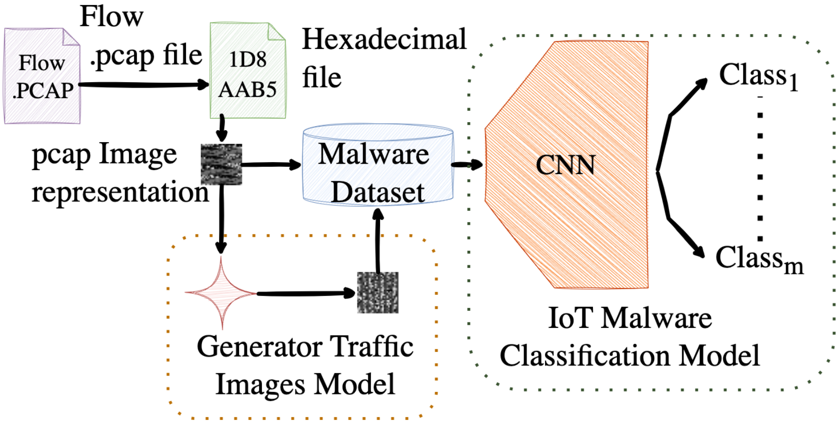


Figure 6.2: The data flow starts with creating hexadecimal-based images for both datasets using the unidirectional flows header. After training the Generative model image-to-image for each class, the dataset is input to the CNN classification model to collect classification outcomes for evaluation.

ally, they are truncated if they exceed 743 bytes, otherwise, zero padding is added, based on [118], were it is shown that the most pertinent information is concentrated in these initial bytes. The resulting files are converted to hexadecimal and then into grayscale images. This choice is coherent with the data parsimony and offers a quick transformation that only needs the header information. In addition, the use of 28x28 px images permits the adoption of lightweight and simple generation and classification models. Then, the generation of image models is initiated to create the synthetic dataset. Hence, the DcGAN, and DDPM are trained on the dataset to achieve superior prediction accuracy. A specific model is considered for each class. Different proportions of synthetic data were set in the final datasets to understand better the relationship between synthetic data and the level of resilience to the lack of data for each method. Considering our study and utilizing the formula 6.1, calculating the β value for the sample size. Let's suppose $F_1 = 0.8$ and $\alpha = 0.05$, then it is possible to proceed with:

$$\beta = \frac{(1+\phi^2) - F_\phi(1+\phi^2+\alpha)}{1+\phi^2+F_\phi} = 0.128 \tag{6.2}$$

Performing the McNemar-Bowker test, the sample size outcome is 735 examples for each class for EDGE-IIOTSET and 580 for the MNT; these differ-

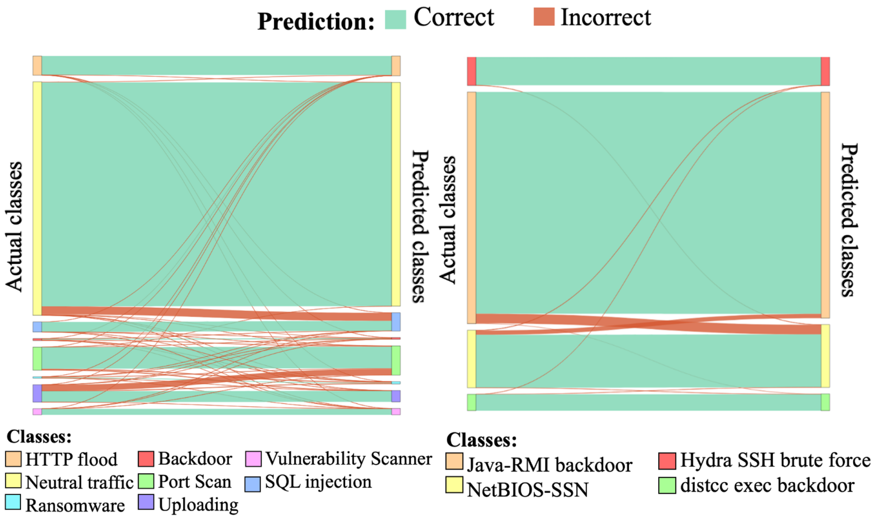


Figure 6.3: The Sankey diagrams visualize the confusion matrix for EDGE-IIOTSET (top) and MNT set (bottom). The critical class to classify for the EDGE-IIOTSET is upload attacks, while for MNT, it's Java RMI backdoor attacks. These diagrams establish the baseline for classification and serve as a reference point for improvement using artificial images.

ences are due to the different number of classes considered. Having defined the datasets, a Convolutional neural network with fixed parameters and hyperparameters was selected to investigate the effect of sample size and the impact of synthetic data on model performance. Also, the baseline model is represented by the same CNN and trained only on the original data, with the sample size defined by our method. The model was trained for 200 epochs, with a learning rate of 0.01 and a stochastic gradient descent optimizer. After the training process, the model's performance in terms of the F1 score is evaluated on the real and unbalanced test sets. This allows for investigating how the performance changes depending on the dataset conformation.

6.4.2 Evaluation of Classification Performance

Figure 6.3 presents a visual description of the resulting confusion matrix for both baseline models in the form of a Sankey diagram [90]. This diagram represents each class with a box on the left or right side, with green arrows for correct classifications and red arrows for misclassifications. This

Table 6.1: F1 scores of the test set with a training model fed with the real dataset and the balanced dataset with a sample size less than the thresholds. The performance in other cases is worse than our method.

num. ex. per classes	**EDGE-IIOTSET**	**MNT**
$\leq$ threshold	0.6	0.7
real unbalanced train set	0.73	0.58
our train set	0.93	0.97

visualization clarifies the unbalanced nature of the datasets and helps to understand the critical classes. For example, in the case of EDGE-IIOTSET, the *Uploading attacks* are the worst classified class, while for the MNT set, the most misclassified class is the *Java-RMI backdoor*. Also, Table 6.1 shows how the performance in terms of F1 score is worse than our baseline in case the number of examples per classes class is minor of the relative threshold or using the real unbalanced train set. **EDGE-IIOTSET.** Focusing on the EDGE-IIOTSET, there are clear differences between the results obtained using GAN-generated images and DDPM-generated images, particularly regarding the classification of neutral traffic. It should be noted that DDPM consistently achieves accurate classification of neutral traffic, regardless of the volume of synthetic data, thereby reducing misclassification variability. Conversely, models trained with GAN synthetic data exhibit increased misclassification, as shown in Figure 6.4. In particular, in these plots, the difficulty of addressing a correct classification for minority classes is evidenced by the curves. For example, in the case of 30%, 50%, and 60% of synthetic data, the DDPM models can not adequately classify the Uploading attacks. In contrast, in the other cases, the results demonstrate that these models can better classify all the critical classes. A motivation for these behaviors can be the increasing noise introduction by the synthetic data and the limited number of epochs, which I set up based on the Baseline experiment. Moreover, Figure 6.5 displays the F1 score for each model along with the relative standard deviation. The baseline model, trained without synthetic data, exhibits high F1 scores and lower standard deviation values. Focusing on the other models, *I find that the DDPM models generally achieve better F1 scores but tend to have higher standard deviations,* except in the case where 30% synthetic data is used. This suggests that while DDPM models can enhance performance, they tend to exhibit greater variability, as said before. Another interesting aspect of these results is the DDPM-based model's capacity to classify better than the GAN-based one's class in the case of 10% synthetic data. However, the overall performance in terms of the F1 score suggests the opposite. **Malicious Network Traffic PCAPs Dataset.** I further analyze the Malicious Network Traffic PCAPs and binary visualization images dataset. Firstly, the main difference between this dataset and the previous one is the smaller number of classes considered. Also, neutral traffic is not considered in this case. However, the ROC curves shown in Figure 6.6 combined with the results in Figure 6.5 confirm the fact that DDPM-based models perform better than the GAN-based models, also for the critical class, both in terms of average F1-scores and their variance. In addition, the performance of both models becomes critical when the percentage of synthetic data is 70%; this implies that the models are confusing for the noising insert from synthetic datasets. This behavior is curios combined with the previous ROC curve, in which the same model topology could not discriminate the classes well when the dataset has 30% of data synthetic. The last comment concerns the GAN-based model trained with 90% of artificial data, where the classification model does not work. To ensure the robustness of our results, I extended the training of the model for an additional 200 epochs, with no significant changes. This indicates that a more complex model is required for this task. In conclusion, the results of the DDPM synthetic images indicate that the dataset building offers a more stable and predictable behavior than that based on GAN images. Additionally, the classification outcomes on the test set with an imbalanced distribution are superior, as are the misclassification confusions as shown in Figure 6.5.

6.4.3 Pixels vs. Packets: Analysis of Synthetic Traffic via XAI

The integration of synthetic traffic data in malware classification models introduces complexities that necessitate a deeper understanding of model behavior. While performance metrics provide an overall measure of the accuracy of the models, they fail to explain how synthetic data influences the decision-making process of the classifier. In our context, explainable AI (XAI) becomes crucial to validate the fidelity of synthetically generated traffic flows by comparing their impact on model decisions against real traffic data. This approach aids in assessing the quality of synthetic data and

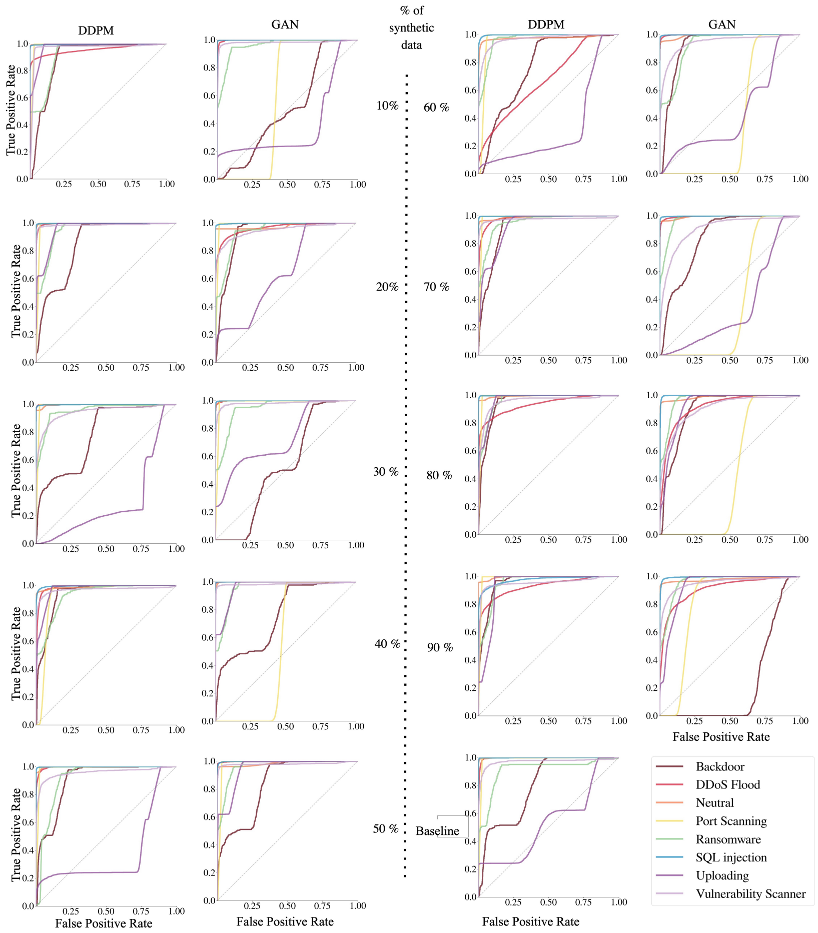


Figure 6.4: ROC curve for EDGE-IIOTSET. These graphs plot the true positive rate against the false positive rate at each threshold setting. DDPM performs better than GAN, as the curve below the bisector line indicates. An increase in misclassification highlights the influence of synthetic data percentage on performance degradation.

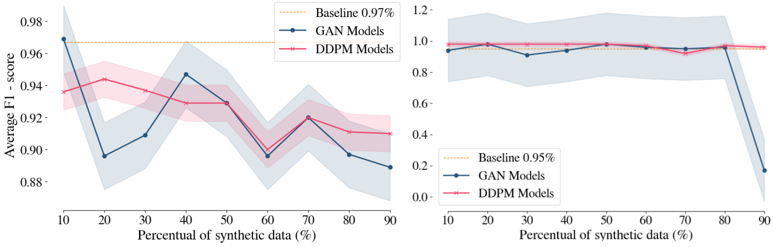


Figure 6.5: Trend of class average F1 score for the test set with a *95% confidence interval.*The EDGE-IIOTSET plot shows that the DDPM-generated data have a 7% higher average F1 score and less variability than the GAN-generated data. The MNT F1 score trend curve highlights that DDPM-based models are more stable with minimal performance variance with more synthetic data. The performance of GAN deteriorates at 90% synthetic data.

also provides insights into potential biases or artifacts introduced by different generation methods. We leverage Gradient-weighted Class Activation Mapping (Grad-CAM) [93], a state-of-the-art XAI technique, to analyze the impact of synthetic traffic generation on classifier processes. This approach allows us to visualize and quantify the salient features driving the CNN's predictions across real and synthetically generated traffic flows.

Grad-CAM Heatmaps. We analyze the generative capabilities of both DDPM and GAN models by exploiting the Grad-CAM heatmaps. Figure 6.7 shows an example of the heatmaps generated by the Grad-CAM algorithm applied to a correctly classified flow sample of the SQL Injection attack. To compare the generative capabilities of DDPM and GAN, we evaluated three CNN-based classifiers trained on different formulations of the EDGE-IIOT dataset: (i) real data only, (ii) DDPM-based data combined with real data, (iii) GAN-based data combined with real data. The heatmaps in Figures 6.7, reveal that the CNN model considers various parts of the traffic flow header with differing levels of importance depending on the input dataset, suggesting the need for a deeper evaluation.

Fidelity Analysis. In this analysis, we aim to deepen the behavior of the classifier by observing how the synthetic traffic data impacts the classification process. We report the distribution of the most influential bytes as explained by the Grad-CAM analysis (Figures 6.8 and 6.9) applied to a CNN-based classifier. Specifically, we map each byte (i.e., a pixel of the flow image) to

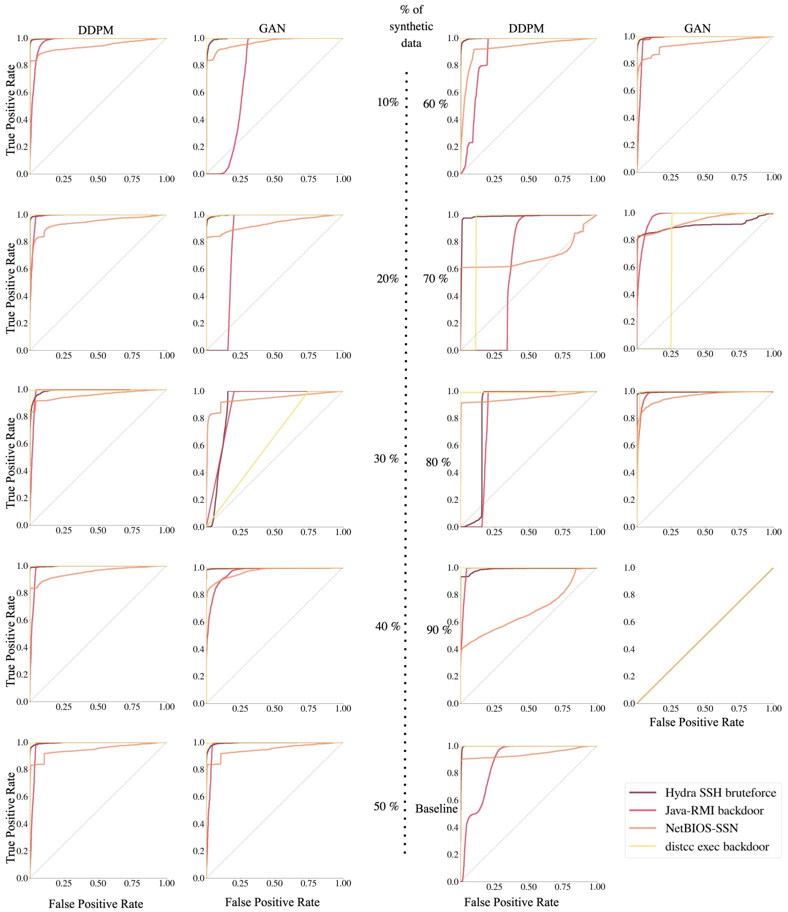


Figure 6.6: ROC curve for the MNT Dataset. The DDPM's performances are better than the GAN's, and their curves appear below the bisector line. The DDPM can more effectively identify and classify different classes. When considering 90% of artificial data, the GAN-based model cannot achieve good classification, while the DDPM model can.

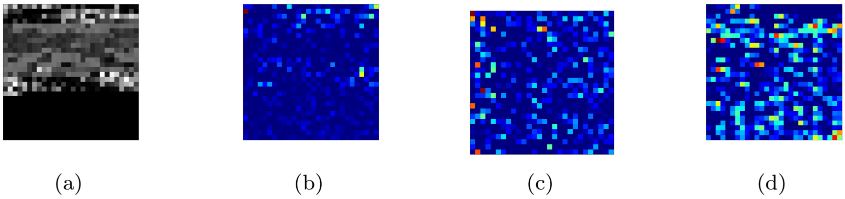


Figure 6.7: Grad-CAM heatmaps of the SQL Injection attack. Red elements indicate a strong influence on the prediction, whereas blue elements have little effect on the prediction. (a) Original image of a SQL injection flow. (b) Resulting Grad-CAM of a CNN model trained only using real data (i.e., no synthetic traffic). (c) Resulting Grad-CAM of a CNN model trained with 50% traffic generated by DDPM. (d) Resulting Grad-CAM of a CNN model trained with 50% traffic generated by GAN.

the corresponding packet header field and select only a group of the most occurring fields.

We can observe interesting differences between the DDPM and DcGAN-generated synthetic traffic. First, the analysis clearly shows that the classifier relies on a different group of header fields depending on how the traffic is generated. While the TCP Acknowledge Number plays an important role for most attacks when data is generated by DDPM, it is not immediate to identify a clear pattern when data is generated by a DcGAN. Moreover, for each generative model, attacks are classified based on different fields. For example, for the DDPM-generated traffic, the classifier relies on a combination of TCP Flags, TCP Window Size, IP header Length, IP Destination, and TCP Sequence Number fields to detect port scanning attacks. While for the GAN-generated traffic, the classifier heavily relies on TCP Flags.

To validate the fidelity of the generated data - i.e., the degree to which the synthetic data resembles the characteristics of the real data it aims to mimic - we compare the distributions of the most influential header fields to the prediction of all attacks between the following use-cases (i) the classifier is trained on DDPM-generated synthetic malware traffic; (ii) the classifier is trained on DcGAN-generated synthetic malware traffic; (iii) the classifier is trained only on real data. Figure 6.11 shows the proportion of each extracted header field over the entire test set. To quantify the similarity

The figure's sub-captions are (a), (b), (c), (d).

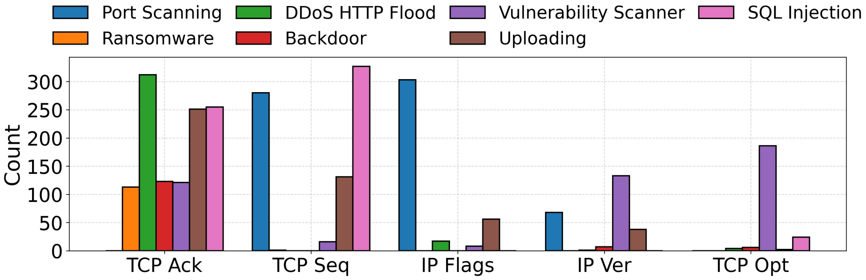


Figure 6.8: Distribution of top-5 header fields across attacks over DDPM-generated synthetic malware traffic.

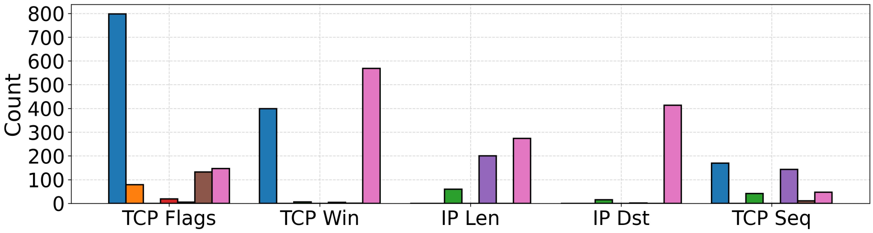


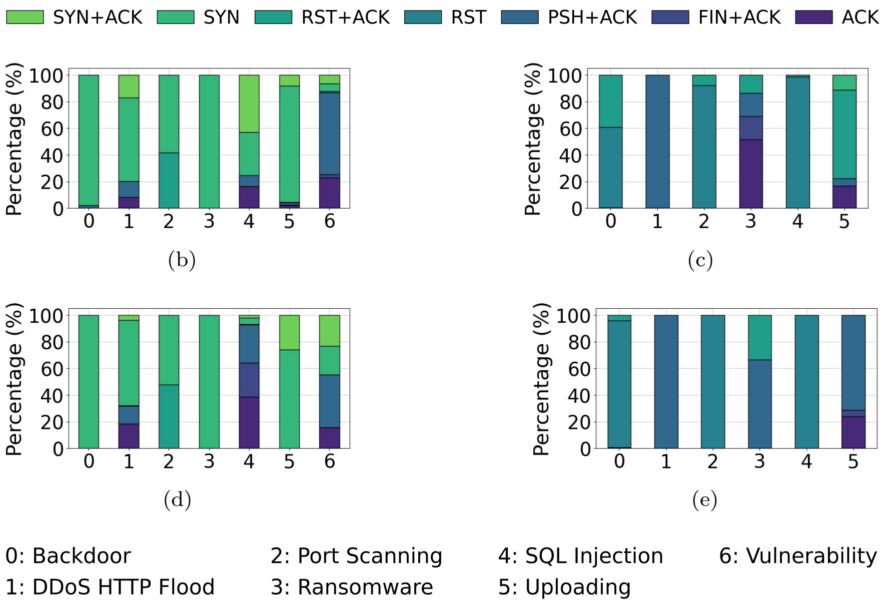
Figure 6.9: Distribution of top-5 header fields across attacks over DcGAN-generated malware traffic.

between the synthetic and real data distributions, we introduce an *overlap score* metric. This score measures the proportion of matching header fields for each prediction between the synthetic and real data. The DDPM model achieves significantly higher overlap scores than the GAN model across all the analyzed header fields, suggesting that the DDPM-generated synthetic data exhibits higher similarity to the real data distribution.

Packet Type Distribution. While header field analysis focuses on the specific values and characteristics of individual packet headers, we provide a complementary analysis by observing which packet type the most influential bytes belong to. By applying the XAI technique, we aim to provide a higher-level view of the overall communication patterns extracted. We further discuss the classifier behavior when trained on synthetic data generated using different generative models. For each most influential byte, we determine the packet type by extracting the TCP flags of the packet header field. In Figure 6.10, we analyze the distribution of the TCP flags for each attack in terms of percentage of occurrence over the predictions corresponding to correct classifications (Figures 6.10b, 6.10d) and misclassifications (Figures 6.10c, 6.10e).

For both generative models, in Figures 6.10b and 6.10d, the classifier relies on SYN packets to identify some of the attacks (i.e., Backdoor, DDoS, Ransomware, Uploading attacks), with a nearly 100% presence in all predictions. Exceptions of this behavior are represented by scanning attacks, where FIN+ACK and ACK packets are considered more relevant, and Port Scanning, which shows a high percentage of RST+ACK flags, and the SQL Injection attack, which shows a mix of SYN+ACK, SYN and ACK packets for the DDPM-generated data, and a mix of ACK, FIN+ACK, and PSH+ACK for the DcGAN-generated data. This analysis shows both generative models' ability to capture the behavior of each attack. When SYN packets are the most influential, they often correspond to attacks that exploit the connection initiation process, e.g., the unusual outbound connection attempts in Backdoor attacks. RST+ACK packets are relevant for the classifier decision in port scanning attacks, i.e. attacks that leverage the normal response of closed ports. Finally, FIN+ACK packets are leveraged by vulnerability scanning to take advantage of less commonly monitored flags to probe systems, and the analysis shows this behavior accordingly.

Interestingly, Figures 6.10c and 6.10e show that the classifier has learned to place undue importance on RST packets when identifying Backdoor, Port Scanning, and SQL Injection attacks, and picks up on patterns in the synthetic data that don't necessarily reflect real-world attack behaviors. One reason is that the classifier may be overfitting to specific patterns in the synthetic data that don't generalize well to real-world scenarios. Moreover, RST packets can be associated with rare abrupt connection terminations that may be disproportionately represented in the synthetic data, due to their inherent rarity. Given these insights, we believe that incorporating domain-specific knowledge about network protocols and attack patterns into both the data generation process and the classifier training can potentially lead to better overall performance. We leave this implementation as future work.



(b)

(c)

(d)

(e)

Figure 6.10: Distribution of TCP flags for packets determined as most influential in the classifier's predictions by the Grad-CAM algorithm. (b) TCP flag distribution for correctly classified DDPM-generated traffic. (c) TCP flag distribution for misclassified DDPM-generated traffic. (d) TCP flag distribution for correctly classified DcGAN-generated traffic. (e) TCP flag distribution for misclassified DcGAN-generated traffic.

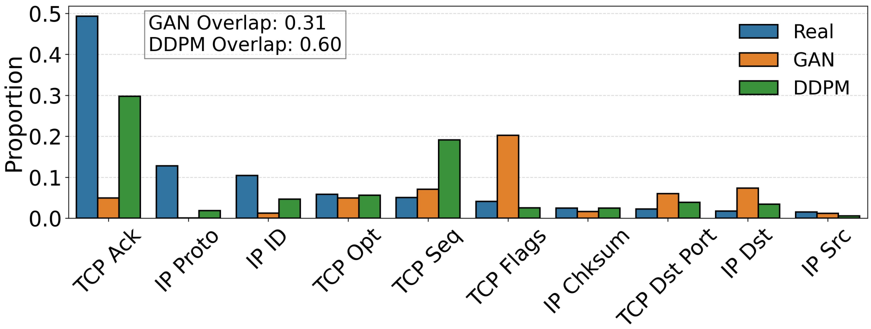


Figure 6.11: Most influential header fields distributions as identified by the Grad-CAM algorithm over three different datasets: real malware traffic images, DDPM-generated, and DcGAN-generated synthetic malware traffic. Overlap scores indicate the similarity between synthetic and real distributions, with DDPM (0.53) achieving higher fidelity than DcGAN (0.27).

Chapter 7

Recognition of Soft Faults: Two Case Studies

The previous chapters focused on the standard and security of IoT networks and their intersection with ML. Indeed, this chapter changes focus and introduces computer vision techniques for the mechatronic industry. In particular, this work wants to explore the potential of image representation in unusual fields such as mechatronic control systems. Thanks to this work, we are beginning to understand the vast potential of sample techniques in contexts that are not commonly found. Also, it could seem that using images is similar to adding complexity. Still, paradoxically, this approach simplifies the calculations and offers a visual representation of the event and the possibility of checking in case of doubts.

The works in this chapter led me to collaborate with colleagues in the smart industry. The first work exploits electrical fault recognition for active magnetic bearings. Unlike classical rolling bearings, these are complex mechatronic systems consisting of mechanical, electrical, and software parts.

The dictionary was built starting from fault signatures consisting of images obtained from the signals available in the system. Subsequently, a convolutional neural network was trained to recognize such fault signature images. This study aimed to develop a fault dictionary and a classifier to identify the most frequent soft elec-

trical faults affecting position sensors and actuators. The proposed method, with its real-time implementation potential, permits a computationally convenient way to determine which component has failed and what kind of failure has occurred. Therefore, this fault identification system determines which countermeasure to adopt to enhance the system's reliability. The performance of this method was assessed through a case study concerning a real turbomachine supported by two active magnetic bearings for the oil and gas field. Seventeen fault classes were considered, and the neural network fault classifier reached 93% of accuracy on the test dataset. [1] [2]

7.1 Introduction

Active magnetic bearings (AMBs) are increasingly used across various rotodynamic applications, ranging from small turbomolecular pumps for medical applications to large compressors in the megawatt range for the oil and gas field. Compared with classic contact bearings, AMBs offer several significant advantages. AMBs significantly reduce friction and the associated wear by levitating the rotor relative to the stator parts. Eliminating friction also enables higher rotation speeds and greater system efficiency and eliminates the need for cumbersome lubrication systems typically required for traditional bearings. Due to the inherently unstable nature of AMBs, a stabilizing feedback control is needed for proper functioning. Therefore, an AMB is a complex mechatronic system comprising various components, including the controller, position sensors, and actuators. The synthesis of the controller is a critical aspect of AMB systems.

For example, various controller structures and design processes can be used, as reviewed in [21, 105]. Among these, augmented PIDs are probably the most widespread choice in industry because of their versatility, accuracy, efficiency, and cost-effectiveness. All the controllers of AMB systems rely on precise measurements of the rotor position, for which a common choice is to use inductive or eddy current sensors, even if recently self-sensing and optical

---

[1] This chapter has been published as "A convolutional neural network for electrical fault recognition in active magnetic bearing systems" in *Sensors,2023, 23.16: 7023.*

[2] This chapter has been published as "A Convolutional Neural Network to Locate Unbalance in Turbomachinery Supported by AMBs" in *2024 IEEE International Workshop on Metrology for Industry 4.0 & IoT (MetroInd4. 0 & IoT), pp. 274-279.*

sensors have also been introduced [110]. The effect of the choice of sensors is discussed, for example, in [18]. Regarding the other components, PWM is commonly used to drive the electromagnets that make the rotor levitate. The behavior of all the components in the loop contributes to determining the stiffness and damping of the bearings, which are directly related to the stability and performance of the closed-loop system. To improve the robustness of such a complex system, a fault detection and diagnostic system is advisable to ensure safe and reliable operation. As a result, several studies have developed methods to detect failures associated with the rotor or with the electrical and electronic components, as, for example, in [33, 92]. Failure detection in AMB systems also enables adopting safety strategies that exploit their closed-loop architecture and adapt to the detected fault. In fact, due to their active nature, controllers can dynamically adjust the bearings' behavior in real-time. An AMB system can enhance its chances of survival and reliability by exploiting the power of internal information processing. For this reason, fault-tolerant AMB systems have been developed to manage malfunctions, for example, using a re-configurable control, as described in [92, 111].

Usually, for fault diagnosis systems, the actual system behavior is compared with the expected one in nominal conditions to identify a faulty system condition. Observers of the system, as described in [33], are typically used.

Another common approach for identifying faults is the simulation-before-test technique. It involves constructing a fault dictionary from simulations of a particular plant, which collects examples of fault signatures that can be used to train a classifier capable of recognizing different faults, as described, for example, in [23].

Traditionally, fault diagnosis has been based on analyzing signals in the time and frequency domains. However, this work proposes an approach that exploits a fault dictionary made by images of signals in the time domain to train a simple convolutional neural network (CNN) that aims to recognize the AMB system's faulty conditions. Taking the available electrical signals as sources, generalized orbits are built and converted into discrete 2D images that are used to fill the fault dictionary with a technique that generalizes and extends the approach proposed by Xunshi, who used the sensors' signals to build orbits to detect mechanical failures [120]. A similar method is proposed by Jing et al., who in their work proposed a feature-based learning and fault diagnosis method for gearbox condition monitoring [66]. The fault features were obtained using a simulation tool developed by some of the authors [17], capable of automatically building the entire fault dictionary once the fault conditions are modeled. In the context of this work, only single electrical parametric faults have been considered for simplicity, but the fault classes can be extended to also include, e.g., mechanical faults, as in [119].

To exploit the knowledge stored in the fault dictionary, this work proposes a classifier based on a convolutional neural network (CNN), well suited for image classification [5, 29], trained with the fault signature examples. In the case of a system fault, the trained convolutional neural network aims to identify the faulty component and the type of fault that has occurred.

Compared with other methods proposed in the literature that rely on analyzing signals in the time domain, the proposed approach, losing the time dependence, offers a novel solution to detect and locate faults, making full use of the potential of smart systems like AMBs. Moreover, with the help of computationally efficient image processing (Adam optimization) and simple neural networks, this automatic online diagnostic system can be developed without excessive computational cost. Such systems could be exploited in advanced prognostic maintenance systems to enhance the balance of plant capabilities over time, as described in [24, 25, 42].

An alternative approach is to employ 1D CNNs, which often exhibit superior performance, as evidenced in [108]. Nevertheless, for this specific study, a traditional CNN architecture was chosen. The primary objective was to develop a tool enabling image interpretation by humans and neural networks. Moreover, the decision to opt for the classic CNN network was influenced by the fact that 1D CNNs treat signals independently until the last layer, partially neglecting the crucial association that characterizes MIMO systems like AMB systems. By using a traditional CNN, these associations are taken into account, making the model more suitable for the specific task at hand.

Another potential approach is the utilization of a Dislocated Time Series CNN (DTS-CNN), which has demonstrated superior performance in non-stationary conditions, as detailed in [70]. However, in the context of this study, only steady-state AMB systems are considered where nearly stationary conditions prevail. Since DTS-CNNs offer advantages in non-stationary settings, their benefits may not be fully realized in this steady-state context, making the classic CNN architecture a better choice for this study.

This chapter collects two case studies on Active Magnetic Bearings (ABM) control systems. In Section 7.2, the electrical fault recognition is presented, while in Section 7.3, the fault location recognition and compensation are exploited.

7.2 Electrical fault recognition:a gentle introduction to the problem

Due to the mechatronic structure of an active magnetic bearing (AMB) system, failures can manifest in various forms, including software, electrical, or mechanical. These faults can result in the high-speed rotor making contact with its housing, potentially compromising the safety of the entire plant. Although touch-down bearings are designed to prevent direct contact between the rotor and housing, such an event must be avoided at all costs. Several measures can be taken to address the diverse range of faults that can arise in AMB systems, including redundancies, quality control, individual measures, and various control strategies. Active fault diagnostics and corrections can also be employed. Guidelines for designing a reliable system are provided in ISO 14,839 [44] and API 617 [74] for turbomachinery on AMBs in the oil and gas field. A three-stage process for dealing with faults is described in [68], which involves determining the timing of the fault, identifying the faulty component, and identifying the type of fault. Given the system's complexity, there are numerous reasons why malfunctions can occur, which can have different degrees of impact on system performance. In [48], the authors summarize the main causes of malfunctions, their occurrence, and severity. This study takes into consideration the most common electrical soft faults, which can be modeled in three different ways: multiplicative, bias, and noise, as discussed in [72, 111]. The multiplicative type of error arises from the failure of system components, which results in changes in gains or sensitivities, commonly affecting amplifiers of conditioning electronics, sensors, power amplifiers, and actuators, usually due to temperature rise, fatigue, or short circuits [111]. Bias faults are related essentially to offset anomalous drifts, e.g., to temperature rise, and noise faults are due to electrical and predominantly impact the position sensors due to the low level of the transduced signals. More specifically, this study considered the most frequent scenario, i.e., that of a single fault at a time. Figure 7.1 provides a summary of the considered faults.

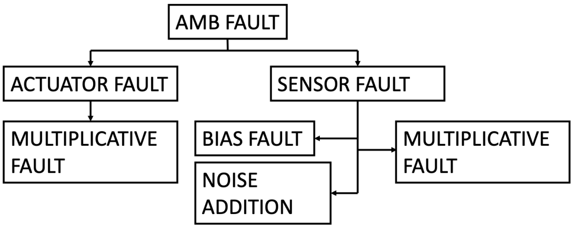


Figure 7.1: Summary of common electrical faults off AMB system.

7.2.1 Fault Dictionary

Steady-state signals obtained from sensors and control systems are exploited to build images that serve as fault signatures. In this study, the available considered signals were position signals from sensors, control signals from the controller, and current signals from the actuators, which are generally easily obtainable in an AMB system. Figure 7.2 summarizes the available electrical signals used to build the dictionary. Since all signals exhibit periodic behavior during normal machine operations at a constant fixed rotational speed, it is possible to create groups of images by representing the signals related to one of the two orthogonal control axes as functions of those related to the other axis, disregarding time as an independent variable and obtaining generalized orbits of the signals related to each bearing, as described in Figure 7.2. The fault features were obtained using a simulation tool to automatically build the entire fault dictionary once the fault conditions were modeled. The simulation tool was implemented in the Matlab-Simulink environment and allowed us to perform Monte Carlo simulations by varying a selected set of component parameters using specific probability distributions or by adding noisy signals. The developed simulation tool was validated by comparing the results obtained with commercial software (MADYN 2000 version 4.5).

Relative to non-faulty conditions, reference images were formed by simulating machine operations at a constant rotor speed and collecting signals while varying the component parameters within their tolerance ranges and accounting for normal noise levels. The dictionary was completed by simulating the different faults that belong to the previously introduced fault classes. In detail, examples of each single soft fault signature in the fault

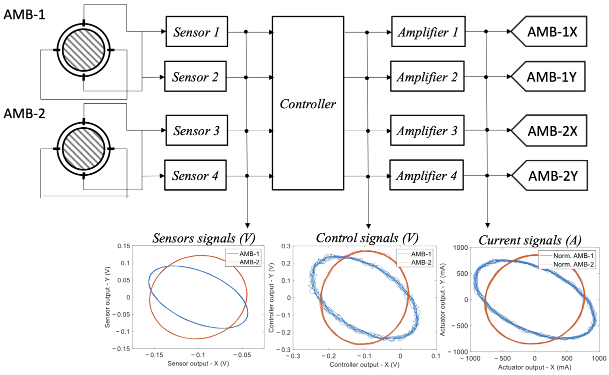


Figure 7.2: Summary of the available electrical signals used to build the dictionary.

classes listed above were obtained by randomly varying all the electrical parameters (gains, sensitivity, and biases) within the tolerance ranges, except for the faulty component, which varied outside this range, or by injecting noise with an abnormal standard deviation.

The variations in the component parameters in the tolerance ranges determine regions of confidence in the reference images relative to non-faulty conditions where the generalized orbits should remain under fault-free conditions. If the orbits go beyond these regions, a fault has occurred. The built fault dictionary consists of images containing the deformed orbits concerning the reference ones obtained by simulating the different faults for the particular plant.

Once the simulations are completed, the feature images can be built. The time signals were normalized, and images with a fixed resolution represented the orbit triplets. Black-and-white images were generated through the Matplotlib library in Python, one for each orbit typology, one for the control signals, one for the position signals, and the last for current signals. Ultimately, these images were concatenated into a unique RGB image so that each RGB channel concerns only one orbit typology. In other words, a unique RGB color was associated with each orbit typology for every triplet of images related to a particular scenario: red for the position signals, blue for control signals, and green for current signals. Directly using images from available signals to train the classifier improves the ability of the system to

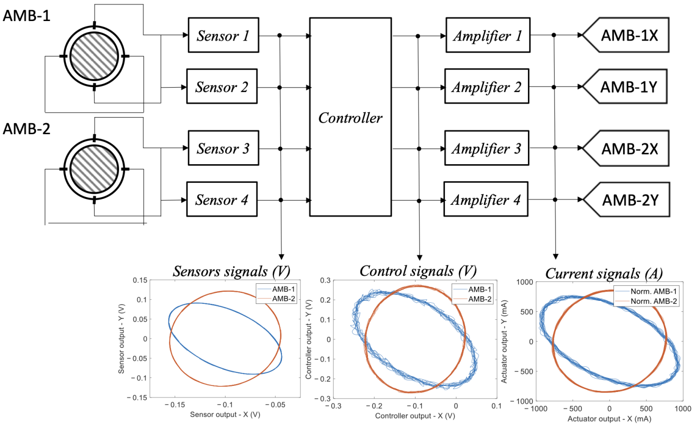


Figure 7.3: Summary of the classes to be classified.

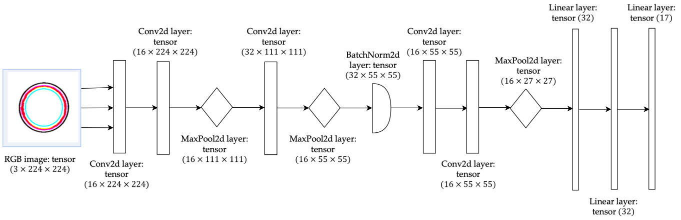

RGB image: tensor (3 × 224 × 224)
Conv2d layer: tensor (16 × 224 × 224)
Conv2d layer: tensor (16 × 224 × 224)
MaxPool2d layer: tensor (16 × 111 × 111)
Conv2d layer: tensor (32 × 111 × 111)
MaxPool2d layer: tensor (16 × 55 × 55)
BatchNorm2d layer: tensor (32 × 55 × 55)
Conv2d layer: tensor (16 × 55 × 55)
Conv2d layer: tensor (16 × 55 × 55)
MaxPool2d layer: tensor (16 × 27 × 27)
Linear layer: tensor (32)
Linear layer: tensor (32)
Linear layer: tensor (17)

Figure 7.4: The proposed model structure.

understand the AMB system state, which offers a wider view and enhances the precision of the information captured.

7.2.2 Case Study

An AMB-supported system was considered as a case study to assess the performance of the proposed diagnosis system. The case study involves a real, medium-sized compressor supported by AMBs for the oil and gas field. Figure 7.5 illustrates the finite element model of the rotor under investigation with a mass of about 810 kg and a length of about 1.80 m. Regarding the electrical part, the controller structure was an augmented PID. It has a decentralized structure that divides the system into control axes, and every control axis is controlled independently. Regarding the other components,

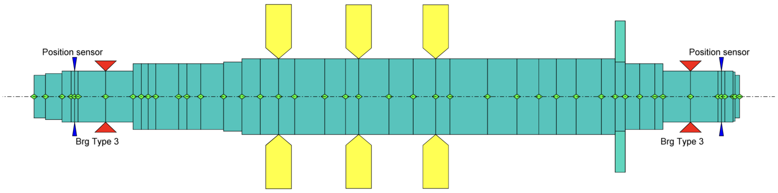


Figure 7.5: This image shows the finite element model of the rotor, where the red triangles are the bearings, the yellow elements are the disks, and the blue elements are the sensors.

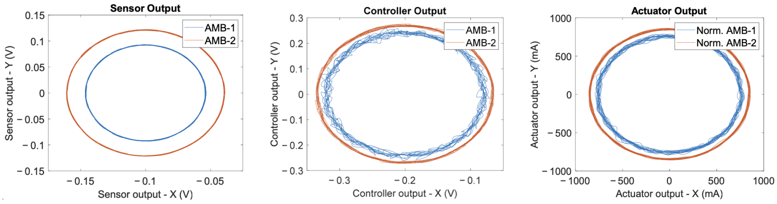


Figure 7.6: This image shows an example of nominal condition orbits. On the left is the sensor output, in the center is the controller output, and on the right is the actuator output.

the system comprises switching pulse width modulation (PWM) amplifiers with heteropolar AMB actuators and inductive position sensors with a band of 4 kHz. Under nominal conditions, the system features a constant rotor rotation speed of 7800 rpm and an imbalance of $2.10\times10^{-3}\ kgm$ positioned at the center of mass of the rotor. The fault-free condition is given by variations in the sensor sensitivities within a 2% tolerance of the sensor, a bias below $1\mu m$, and in the presence of Gaussian white noise with a standard deviation of 1 μm. As for the actuators, a 2% tolerance of the DC gain is taken into consideration. Figure 7.6 displays an example of nominal condition orbits related to the fault-free signatures for the two radial bearings of the rotor.

The faulty conditions that were considered to form the fault dictionary classes were computed through Monte Carlo analysis using specific parameter distribution. Specifically, for each actuator multiplicative fault, the faulty actuator gain was chosen as a uniformly distributed random number out of the tolerance range centered in the nominal value in an interval of ±50% of

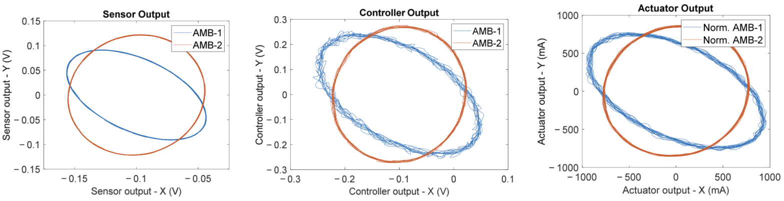


Figure 7.7: Example of signal orbits related to a sensor multiplicative fault, specifically an AMB-1x-axis multiplicative fault of −50% with respect to the nominal value.

the nominal value. Similarly, for each sensor multiplicative fault, the faulty sensor gain was chosen as a uniformly distributed random number out of the tolerance range centered in the nominal value in an interval of ±50% of the nominal value. For each sensor noise fault, the faulty sensor was subjected to a Gaussian additive noise with a standard deviation up to five times the nominal level of noise. Finally, for each sensor bias fault, the faulty sensor had a random bias amplitude larger than up to five times the nominal level of bias. Figure 7.7 shows an example of how orbits change shape when a specific fault occurs with respect to the orbits related to the nominal conditions reported, for example, in Figure 7.6. The proposed CNN model consists of a series of alternately stacked convolutional, pooling, and fully connected layers, as shown in Figure 7.4. It employs convolutional layers to extract features, max pooling to down-sample the feature maps, and fully connected layers to map the extracted features to the output classes. The convolutional layers play a key role in feature extraction, which is a critical step in the model's learning process. The first layer was a convolutional 2D layer, which had the images of orbit triplets as input channels, while the output channels were the 17 classes. In particular, 16 are fault classes, and one is the nominal class related to normal operating conditions. These classes, which include various fault types and normal conditions, are summarized in Figure 7.3 and refer to Figure 7.1.

The dataset used in the case study consists of 37,600 RGB images obtained from different simulations, as described above in Section 7.2. Specifically, each image has a size of $3 \times 224 \times 224$, where 3 represents the RGB channels and 224×224 is the number of pixels used for each image. Figure 7.8 gives an example of the RGB images used.

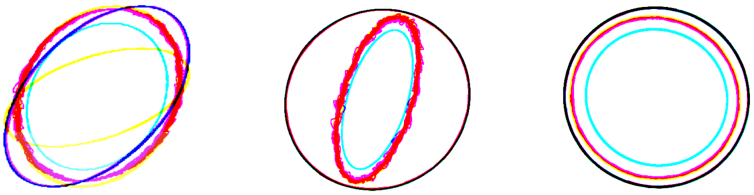


Figure 7.8: The RGB images used for training. The right and central panels show the actuator and sensor gain faults. In the panel on the left, the nominal condition is shown.

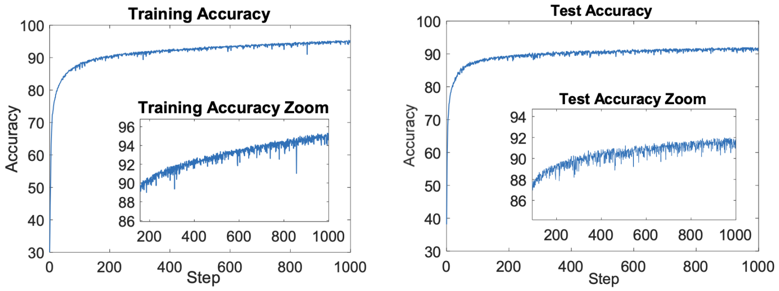


Figure 7.9: On the left, the plot of the training accuracy for each epoch is shown; on the right, the test accuracy for each epoch is reported.

7.2.3 Results and Discussion

The dataset, obtained as described in the previous section, comprises 5600 examples of the fault-free condition and 2000 examples of all the other 16 fault classes. It was divided into a training set, which includes 80% of the examples, and test and validation sets, each of which includes 10% of the remaining examples.

The model was trained using the pre-processed training set for a total of 1000 epochs. The time required for the training process was about seven hours using a GPU NVIDIA RTX A6000 48 GB GDDR6. Using the test set, the proposed CNN was validated; in fact, the test set accuracy reached was about 93%, while the training set accuracy reached was about 95%. These results are presented in Figure 7.9, with the test set accuracy depicted in orange and the training set performance in pink.

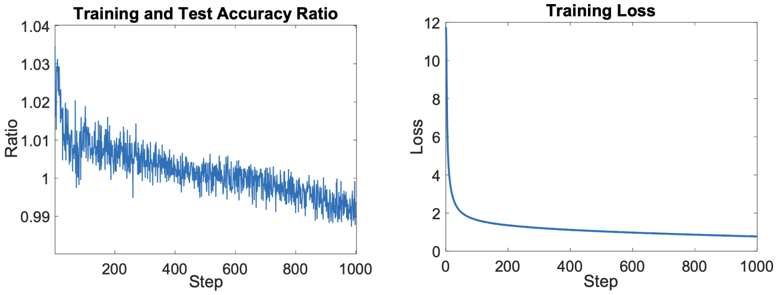


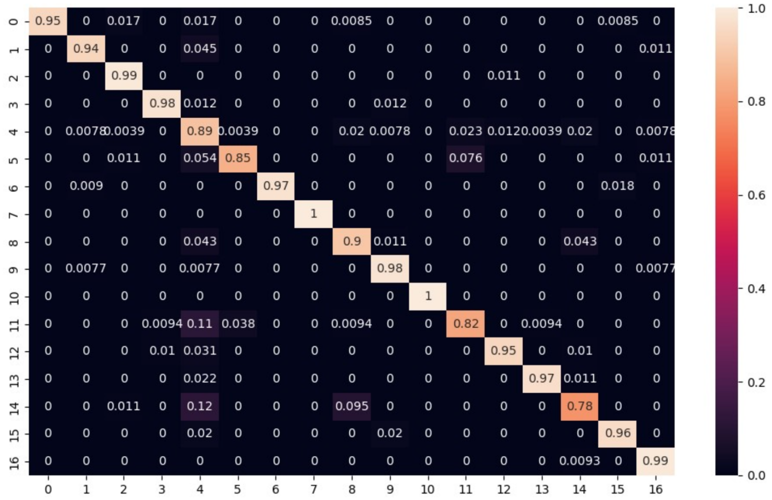
Figure 7.10: On the left, the training and test accuracy ratio plot is shown, and on the right, the training loss plot is reported.

Figure 7.10 shows the test and accuracy ratio, which represents the ratio between the accuracy of the training and the test reported in Figure 7.9. The descending trend observed in this plot suggests that overfitting does not affect CNN's estimated parameters. This indicates that the model has achieved a good level of generalization, meaning it can effectively generalize its learned patterns and make accurate predictions on unseen data.

Additionally, the obtained confusion matrix is reported in Figure 7.11. The horizontal axis of the matrix represents the actual faults, while the vertical axis represents the predicted faults. The diagonal elements show the percentage of correct predictions; the values above the diagonal indicate false positives, while the lower ones represent false negatives.

As shown by Figure 7.11, the confusion matrix is quite diagonal even though the lowest values are about 70%. Most of the misclassifications were found to be related to soft faults related to small parametric deviations (with respect to the fault-free condition).

The achieved accuracy is quite satisfactory for soft fault recognition taking into account small parametric variations as well. This was due to two reasons. The first is that the regions of parameter variation related to faults border the tolerance regions. The second reason is that the image resolution is limited. These facts led some classes not to be recognized correctly, specifically the faulty classes related to orbits that differ by a small amount for the reference ones in the presence of a small variation in a parameter out of the tolerance ranges. This is a problem intrinsic to the definition of a soft fault itself. No net borders between the faulty and the fault-free conditions



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0	0.95	0	0.017	0	0.017	0	0	0	0.0085	0	0	0	0	0	0	0.0085	0
1	0	0.94	0	0	0.045	0	0	0	0	0	0	0	0	0	0	0	0.011
2	0	0	0.99	0	0	0	0	0	0	0	0	0	0.011	0	0	0	0
3	0	0	0	0.98	0.012	0	0	0	0	0.012	0	0	0	0	0	0	0
4	0	0.0078	0.0039	0	0.89	0.0039	0	0	0.02	0.0078	0	0.023	0.012	0.0039	0.02	0	0.0078
5	0	0	0.011	0	0.054	0.85	0	0	0	0	0	0.076	0	0	0	0	0.011
6	0	0.009	0	0	0	0	0.97	0	0	0	0	0	0	0	0	0.018	0
7	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0.043	0	0	0	0.9	0.011	0	0	0	0	0.043	0	0
9	0	0.0077	0	0	0.0077	0	0	0	0	0.98	0	0	0	0	0	0	0.0077
10	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
11	0	0	0	0.0094	0.11	0.038	0	0	0.0094	0	0	0.82	0	0.0094	0	0	0
12	0	0	0	0.01	0.031	0	0	0	0	0	0	0	0.95	0	0.01	0	0
13	0	0	0	0	0.022	0	0	0	0	0	0	0	0	0.97	0.011	0	0
14	0	0	0.011	0	0.12	0	0	0	0.095	0	0	0	0	0	0.78	0	0
15	0	0	0	0	0.02	0	0	0	0	0.02	0	0	0	0	0	0.96	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.0093	0	0.99

Figure 7.11: Confusion matrix of test set.

exist, but in any case, false positives or negatives related to these borderline situations have no severe consequences.

7.3 Fault location recognition: a gentle introduction on problem

Due to the complex mechatronics nature of AMB systems, failures can manifest in different ways, such as software, electrical, or mechanical, which can impact system performance differently. Lijesh *et al.* in [74] summarizes common malfunction causes, their occurrence, and severity. These faults can lead to high-speed rotor contact with the housing, exposing plant safety. Measures like redundancies, quality control, individual actions, and various control strategies are essential to mitigate these risks. Active fault diagnostics and corrections are also valuable. This work focuses on unbalance faults. A residual imbalance always remains even if the rotor is balanced during manufacturing. Following an identification process, this imbalance is measured, and the particular turbomachine is characterized. Due to AMB levitation, wear mainly occurs on the impeller blades during normal operation. Solid particle erosion is one of the main reasons for blade wear failure.

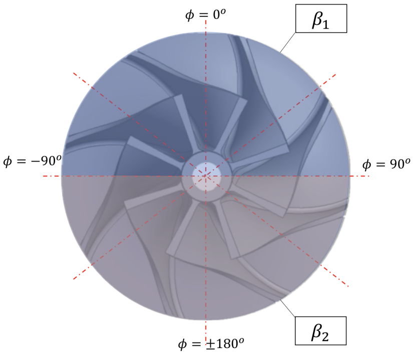


Figure 7.12: Impeller sector division.

Wang *et al.* in [116] analyzed the primary influencing factors on erosion, such as the particle characteristics, environmental condition, and materials characteristics. Normally, the worn blades must be localized and repaired in an impeller. Every worn blade acts as an unbalanced fault that contributes to the vibrations of the entire AMB closed-loop system.

This work aims to localize the faulty impeller and the position of the faulty blade, considering a single fault at a time. More specifically, the localization is intended to identify the phase ϕ between the native unbalance ($\phi = 0$) of the particular turbomachine and the faulty blade. To this aim, the impeller is divided into equal circular sectors, and this work aims to locate the sector that contains the faulty blade. Fig. 7.12 summarizes this idea. In this work, for demonstration purposes, only two circular sectors (β_1 and β_2) per impeller were considered, but the sector number can be arbitrarily increased if needed.

7.3.1 Fault Dictionary

Steady-state signals are exploited to construct images that serve as fault signatures, which act as input data to train a classifier. In this work position, signals from sensors are considered, and they are always available in an AMB system. Considering that all signals inherently present periodic behavior when the turbomachine operates at a fixed rotational speed under steady conditions, it becomes feasible to organize these time signals into couples by expressing the signal associated with one of the two orthogonal control axes of a radial AMB as a function of the other. Time is no longer treated as an independent variable in this process, and one generalized signal trajectory or orbit is generated for each radial bearing. These orbits change in the presence of a fault, assuming specific shapes corresponding to the related fault. Therefore, collecting the different orbit signatures in the presence of different faults can be used as a fault dictionary.

The fault orbits were derived through a simulation tool that automatically builds the entire fault dictionary modeling the fault conditions, as described in [17]. This simulation tool has been implemented within the Matlab-Simulink environment. It enables the accomplishment of Monte Carlo simulations by using specific probability distributions to simulate the parameter variation ranges of the AMB system components or by adding noisy signals. The developed simulation tool was validated by comparing the results obtained with commercial software (MADYN 2000). Machine operations at a constant rotor speed were simulated while varying the component parameters within their specified tolerance ranges and accounting for normal noise levels to create reference orbits linked to the non-faulty condition. The dictionary is completed by collecting the orbit orbits linked to faulty conditions obtained by simulating the different impellers' unbalanced faults by randomly varying different unbalanced phases and magnitudes. Variations within component parameter tolerance ranges define confidence regions containing the reference orbits representing the fault-free condition. Any deviations of the orbits outside these regions signify a fault occurrence. Images with a fixed resolution represented the orbit signatures. Black and white images were generated using Python's Matplotlib library. Utilizing images directly from the available signals to train a classifier enhances the system's capability to comprehend the AMB system state. This approach provides a broader perspective and improves the precision of captured information.

7.3.2 Case study

As a case study to assess the performance of the proposed diagnosis method, a medium-sized expander compressor supported by AMBs, the same as described in the last case study.

For this case study, tolerance parameters encompassed a 2% variation in sensor sensitivities, sensor biases of 1 μm, and Gaussian white noise characterized by a standard deviation of 1 μm. Regarding the actuators, a 2%

Classes name	Class size
Nominal Condition	3000
Expander fault in β_1	2000
Expander fault in β_2	2000
Compressor fault in β_1	2000
Compressor fault in β_2	2000

Table 7.1: Classes distribution

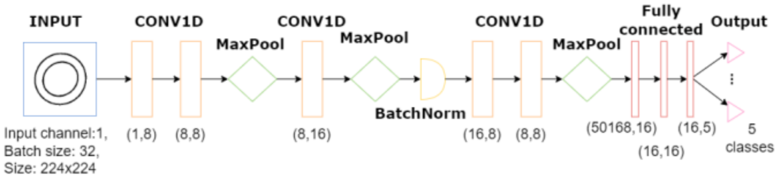


Figure 7.13: CNN proposed model structure, the numbers below the layer represent the input and output channels.

tolerance of the DC gain was considered. Since the number of impellers of the system is two, the fault classes considered were four, considering two circular sectors per impeller ($\beta_1$and β_2), as shown in Fig. 7.12. As anticipated, the CNN classifier is designed to identify five classes: the Nominal Condition, two unbalanced faults on the expander impeller, and two unbalanced faults on the compressor impeller summarized in Table 7.1. The classifier structure is illustrated in Fig. 7.13. The faulty conditions that formed the fault dictionary classes were simulated with a Monte Carlo analysis by varying the fault unbalance magnitude in the range $[0.6.00 \times 10^{-4}]$ kgm and its phase in the prescribed range using a uniform distribution.

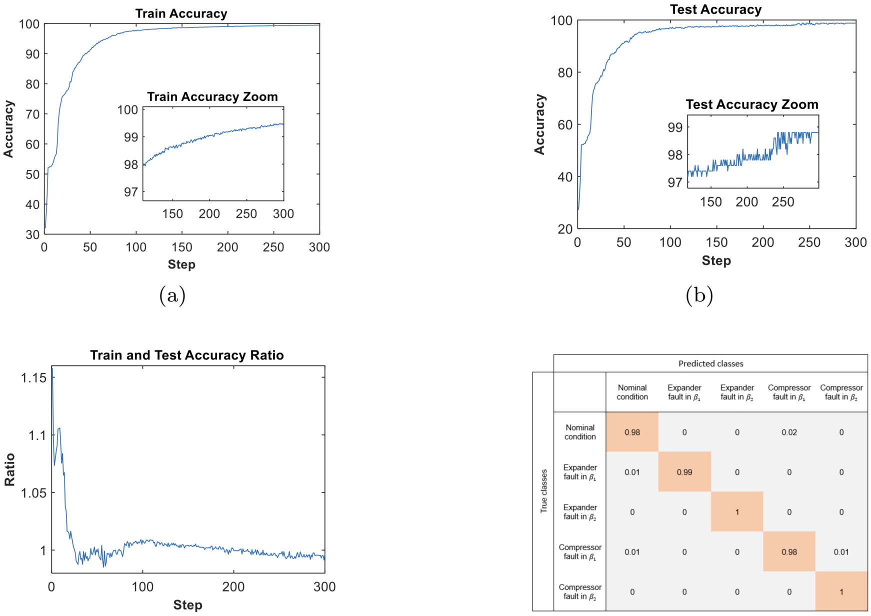
7.3.3 Result and comments

The dataset, obtained as described in the previous section, comprises 3000 examples of the fault-free condition and 2000 examples for each of the other 4 fault classes. The training set includes 80% of the examples, and the test and validation set collects 10 % of the remaining examples each. Each image had a size of 224 × 224 pixel.

The model was trained using the pre-processed training set (a batch size of 32) for 300 epochs, using a learning rate of 10^{-6}. This training took approximately four hours, utilizing a GPU NVIDIA RTX A6000 with 48 GB GDDR6. The proposed Convolutional Neural Network (CNN) was validated using the test dataset, where it achieved an accuracy of approximately 98% . Simultaneously, the training dataset yielded an accuracy of around 99%. The obtained interference time is 2×10^{-4}s, validating the feasibility of implementing the proposed model in real-time applications. These results are graphically represented in Fig. 7.14.a and Fig. 7.14.b, showcasing test set accuracy and training set performance, respectively.

Fig. 7.14.c illustrates the test and accuracy ratio, which shows the relationship between the training and test accuracies depicted in Fig. 7.14.a and Fig. 7.14.b. The declining trend observed in this graph suggests that the estimated parameters of the CNN remain unaffected by overfitting. This observation implies that the model has achieved a good level of generalization, indicating its capability to effectively apply learned patterns and make accurate predictions when confronted with previously unseen data.

Furthermore, Fig. 7.14.d presents the resulting confusion matrix, in which the horizontal axis represents the actual faults, and the vertical axis represents the predicted faults. The diagonal elements represent the percentage of accurate predictions, while values above the diagonal denote false positives, and those below signify false negatives. As depicted in Fig. 7.14.d, the confusion matrix displays a notably diagonal pattern, with even the lowest values hovering around 98%. Most misclassifications are primarily associated with minor parametric deviations from the fault-free condition. The accuracy of recognizing these faults, even with small parametric variations, is quite satisfactory for two reasons. First, the regions of parameter variation related to faults closely border the tolerance regions. Second, the limited image resolution contributes to some classes being recognized less accurately, specifically those associated with orbits that exhibit minor differences compared to the reference orbits when subjected to slight parameter variations beyond the tolerance, as there are no clear-cut boundaries between faulty and fault-free conditions. However, it's worth noting that false positives or negatives in these borderline situations do not have significant consequences. Similar accuracy was also achieved in [36], where a 3D convolutional neural network was implemented to identify electrical faults, and in [58], where several convolutional neural networks were tested to identify simpler mechanical



(a)

(b)

(c)

True classes	Predicted classes				
	Nominal condition	Expander fault in β_1	Expander fault in β_2	Compressor fault in β_1	Compressor fault in β_2
Nominal condition	0.98	0	0	0.02	0
Expander fault in β_1	0.01	0.99	0	0	0
Expander fault in β_2	0	0	1	0	0
Compressor fault in β_1	0.01	0	0	0.98	0.01
Compressor fault in β_2	0	0	0	0	1

(d)

Figure 7.14: (a) Train accuracy for each epoch. (b) Test accuracy for each epoch. (c) Train and test accuracy ratio plot. (d)Confusion matrix of the test set.

faults.

Chapter 8

Conclusion

In conclusion, this thesis has thoroughly examined the complexities associated with the proliferation of Internet of Things devices and their management. Analyzing existing platforms revealed critical requirements for interoperability, scalability, and effective governance, which are often inadequately addressed by mainstream solutions that struggle to integrate legacy systems. The introduction of the IoT Directory, paired with automated reasoning tools, represents a significant advancement in the field, offering enhanced flexibility and adaptability within the open-source Snap4City platform.

Additionally, the research has tackled the pressing issue of security vulnerabilities inherent in decentralized IoT ecosystems. The study demonstrated that federated models, particularly those utilizing global standardization and Principal Component Analysis, outperform traditional centralized methods in precision-recall metrics by employing a federated learning approach. The findings underscore the potential of the Federated Average method to deliver optimal performance while minimizing traffic overhead, thereby enhancing the scalability of IoT security applications.

Moreover, advancements in image classification for network traffic analysis, coupled with innovative artificial data generation techniques, have shown promise in enhancing the classification of malware traffic, even in data-scarce environments. The proposed fault diagnosis method for Active Magnetic Bearings exemplifies the research's practical applications, demonstrating adaptability and accuracy in identifying system faults.

Future Research direction. Future studies in the Internet of Things should prioritize enhancing interoperability among diverse platforms through the development of standardized ontologies and semantic frameworks. Such advancements are vital for creating a cohesive IoT environment that adapts to various system architectures.

The incorporation of AI-driven anomaly detection methods within IoT ecosystems is also essential for proactive security measures. This will enable real-time identification and mitigation of potential threats. Furthermore, exploring edge AI techniques will distribute intelligence closer to the source, significantly reducing latency and improving overall system efficiency. This might also leverage standard protocols like Manufacturer Usage Description (MUD) [71]

Additionally, hybrid learning approaches that combine federated learning with transfer learning could enhance model generalization across diverse IoT environments, yielding robust models capable of performing well under dynamic conditions. Finally, expanding fault diagnosis methods to include predictive maintenance strategies will bolster the reliability and lifespan of critical industrial systems, ensuring sustained functionality in an increasingly interconnected world.

Appendix A

Publications

This research activity has led to several publications in international journals and conferences. These are summarized below.[1]

International Journals

1. Donati, Basso, Manduzio, Mugnaini, Pecorella, *Camerota*. "A Convolutional Neural Network for Electrical Fault Recognition in Active Magnetic Bearing Systems." Sensors 2023, 23, 7023. [DOI: 10.3390/s23167023] *6 citations*

International Conferences and Workshops

1. *Camerota*, Pappone, Esposito, Pecorella "Addressing Data Security in IoT: Minimum Sample Size and Denoising Diffusion Models for Improved Malware Detection", in *International Conference on Network and Service Management*, Prague (Czech Republic), 2024
2. Bellini, **Camerota**, Nesi "Automating Heterogeneous IoT Device Networks from Multiple Brokers with Multiple Data Models.", in *Global IoT Summit.*, Dublin (Ireland), 2022.
3. *Camerota*, Pecorella, Bagdanov. "The intrinsic convenience of federated learning in malware IoT detection", in *Workshop on Wireless Networks Deployment and Management* , Prague (Czech Republic), 2024
4. Donati, *Camerota*, Basso, Mugnaini, "A Convolutional Neural Network to Locate Unbalance in Rotors Supported by AMBs ", in *I2MTC*, 2024

---

[1] The author's bibliometric indices are the following: H-index = X, total number of citations = XX (source: Google Scholar on Month XX, 201x).

Automating Heterogeneous Internet of Things Device Networks from Multiple Brokers with Multiple Data Models

Pierfrancesco Bellini
Distributed Systems and Internet Technology
Lab DISIT, Florence, Italy
0000-0002-8167-1003

Chiara Camerota
Distributed Systems and Internet Technology
Lab DISIT, Florence, Italy
0000-0002-0363-2404

Paolo Nesi
Distributed Systems and Internet Technology
Lab DISIT, Florence, Italy
Paolo.nesi@unifi.it ,
https://www.snap4city.org
0000-0003-1044-3107

Abstract — **The Internet of Things (IoT) is becoming pervasive and with each new installation of the IoT platform legacy internal and external brokers have to be integrated. Internal brokers are those under the control of the platform, while external brokers are managed by third parties. Both brokers kind may be multiservice / multi-tenant and may manage multiple Data Models. The interoperable management of these complex networks has to pass from the IoT device registration which is typically a re-current operation since the IoT networks are in continuous evolution. In this paper, the above-mentioned problems have been addressed by the introduction of our concept of IoT Directory and reasoning tools to (i) manage Internal and External brokers, (ii) perform the automated registration by harvesting and reasoning of devices managed into external brokers single- or multi-tenant services, (iii) perform the automated registration and management of Data Models, and any custom Data Model. The solution has been developed and tested into Snap4City, an 100% open-source IoT platform for Smart Cities and Industry 4.0, the official FIWARE platform, EOSC, and lib of Node-RED. The specific IoT Directory has been developed in the context of the Herit-Data Project, the results have been validated in wide conditions of the whole Snap4City network of more than 18 tenants, and billions of data.**

Keywords— Automated IoT (Internet of Things) Device Registration, Internal and External IoT (Internet of Things) Brokers, IoT (Internet of Things) Network, Smart Data model, Snap4city

I. INTRODUCTION

The Internet of Things (IoT) defines a paradigm for the computation and communication among things that everyone uses more and more daily [1]. It is due to the intense deployment campaign worldwide about Low-Power Wide Area Network technologies [2]. Nowadays, the real world and IoT devices are integrated tother with some humans in the loop. Communication among devices may support various protocols (e.g., Message Queue Telemetry Transport or MQTT, Next Generation Service Interfaces or NGSI, Advanced Message Queuing Protocol or AMQP, Constrained Application Protocol or CoaP) thus, the cloud-fog infrastructures are exploited, as well as the management of the information [3]. In terms of data management, the complexity is growing, not only for the huge amount of data but also for the need of interoperability and abstraction. The **Gateway** concept is a relevant entity to manage IoT devices into an **IoT Platform**. It may be integrated with one or several **IoT Brokers** to send/receive data to/from devices. The Gateways and its IoT Brokers are typically based on a single protocol and managed by third parties as a public service for several customers interested in the same area (for example LoraWAN services [4]). A Gateway may abstract from the IoT Broker level managing them for multiple organizations/tenants (which can be regarded as customers of Gateway services to manage a number of **IoT Devices**), via some API and/or Web user interface. Typical IoT Brokers are capable to manage only one organization, and thus they are single-tenant, in the sense that they broker messages using the topic concepts (which can be regarded as the key for subscription) without any internal partition of services as a sort of family of devices and subscriptions. Some IoT Brokers can be multi-tenant, such as the FIWARE **Orion Broker**, which provides a partition of the devices in groups, and each of them may have a dedicated service/path for a specific scope (or of a specific customer). Furthermore, devices of different tenants could exist physically in different places (even having identical IDs), and the subscription to the broker's tenant may imply getting all messages/services in the partition. That is feasible only if the subscriber knows the identifier of the service path and, in the cases of access control, has the grant to access at the services. The complexity of **IoT Platforms** grows with the need of managing multiple IoT Brokers which can be managed by third parties different from the IoT Platform manager, i.e., **IoT External Brokers**, and/or directly managed by the IoT Platform-tools, i.e., **IoT Internal Brokers**, can be adopting different protocols, formats.

For the **IoT External Brokers**, the entities (IoT Devices) are directly registered on the IoT Broker which is not under the control of the IoT Platform. Thus, the IoT Platform does not know the IoT Device data structure nor the composition of messages and services. Most of the IoT Platforms neglect these interoperability and integration aspects and provide simplified solutions. They do not care about Internal/External Brokers, just providing the possibility to set up end-to-end solutions with some restricted usage, for example using only internal brokers they provide. Thus, AWS IoT by Amazon (AWS) [5] and Siemens MindSphere [6] make the broker structure transparent to their users unless they buy a specific add-on. While IoT Platform like Google IoT Cloud (Google IOT) [7] shows the Broker architecture but allows the usage of only one kind of protocol (e.g., MQTT). At least, solutions like MS Azure IoT (MS Azure) [8] or IBM Watson [9] are more flexible. MS Azure does not provide to cluster their objects, in other words, supporting only one organization on broker; the IBM solution does not allow the connection of External Brokers. In summary, most of the solutions provide simple scenario, and mainly assume to have customers starting to use their solution from scratch (on cloud or on premise), offering limited capabilities to deeply integrate the platform with legacy IoT Broker and Network. Nevertheless, all of them provide the possibility of connecting to other IoT Brokers and Network by means of REST Call on API. Naturally, in those scenarios, the third-party brokers are not directly connected and neither managed in terms of IoT Device registration, subscription, data storage, search, etc.

Different IoT Devices connected to the same broker adopt the same protocol and may use different data models. If the message format is based on JSON, the corresponding schema may be defined and used for validation, while variables/attributes can be differently defined. For example, FIWARE Orion Broker adopts the NGSI protocol with the possibility of managing the so-called **Smart Data Models** from which IoT Devices can be templated out [10]. The IoT Platform or Gateway must recognize these **IoT Device Models** and manage them (registering, processing, producing, storage, etc.). In the case of **IoT External Brokers**, the IoT Platform may not know the IoT device models, and neither the identifier (topic) of the External IoT Devices. Hence, at the arrival of a message from an unknown device (which can partially provide information in its body, typically not the metadata, since most of the devices minimize the data transmission), the IoT Platform is not in the condition of registering the device, and neither to correct link the message to the former devices. Thus, the devices and messages are easily managed when a new external device is added to the IoT Platform if the data model adopted is known. In other words, if the IoT Platform knows the data model adopted by a device it is easier to identify the data structure and so it could automatically manage the relationships among data entities and verify coherence. For all these reasons, the Data Model concepts and formalisms are crucial.

In this paper, the above-mentioned problems and other aspects have been addressed to design and implement a solution for leveraging IoT network interoperability and the management of Data Models. To this end, we have created the concept and tool named **IoT Directory** in the Snap4City architecture [11]. The **IoT Directory** supports: (i) **Internal** and **External** brokers, (ii) the automated registration of devices managed into External Brokers single- or multi-tenant services, (iii) automated registration by harvesting and reasoning of IoT Devices compliant with standard models such as FIWARE Smart Data Model, and any custom Data Model in Snap4City IoT Device Model providing a formal semantic definition of attributes. The research presented in this paper has been developed for Snap4City architecture presently quite diffuse in Europe as 100% open source IoT platform for Smart Cities and Industry 4.0 and it is an official FIWARE platform and solution, and of EOSC, Node-RED [12], [13]. The specific IoT Directory has been developed in the context of Herit-Data Project which promotes the use of smart and open data to better manage tourism flows in natural and cultural heritage sites, the results have been validated in wide condition of the whole Snap4City network of more than 18 tenant, and billions of data.

The paper is organized as follows. Section 2 presents the major requirements for data model and broker interoperability in IoT Platforms. Section 3 shows the role of IoT Directory in IoT architecture for managing internal/external brokers with the aim of automated registration, management vs data models and formal definition of their attributes. In Section 4, some details regarding the validation of the solution are reported. The validation included verifying the processing timing and giving a general numeric KPI about the shape of the entities. In Section 5, the conclusions are drawn.

II. REQUIREMENTS ANALYSIS AND RELATED WORK

In this section, the requirements that an IoT Platform for IoT Network management and exploitation should satisfy are reported and commented on. They have been identified in the context of workshops, conventions, and analysis for the development and exploitation of the Snap4City platform covering smart city and Industry 4.0 domains. The requirements for the IoT Platform are presented in logical order from R1 to R10 as follows. The following list of the requirement refer also to a set of well-known platforms: AWS, Google IOT, MS Azure, Siemens MindSphere and IBM Waston. Therefore, an **IoT Platform** should provide support to:

1. **Manage different kinds of IoT Brokers, IoT Devices and IoT Edge Devices**. They should be based on different protocols, formats, and modalities to establish connections with the IoT Platform. Focusing on the Platform considered, all of them support MQTT and HTTP, while Google IoT and Azure IoT support only MQTT Broker. It is important to highlight that most of the platforms provide specific components for different protocols, for instance: Amazon MQ that supports Broker with AMPQP, MQTT, OpenWire and STOMP protocol.
2. **Connect External and Internal Brokers.** They could be multiservice and could provide different protocols. Internal Brokers should be deployed and registered by the IoT Platform, while the External Brokers would be only registered to use them. In the Platform considered, the brokers are the products' core of stakeholders offers, for this reason the requirement is partially satisfied. In that sense, AWS IoT and Siemems MindSphere offer a paid add-on.
3. **Register, manage and use messages conformant to any Data Model with any data type**. Providing, receiving, managing, storing, and retrieving messages for any IoT Device of any Data Model with its attributes and data types, and related access control. A Data Model provides a model format for IoT Device messages with several variables/parameters/attributes with their specific data types. For the listed Platforms, the messages from the IoT Devices are freely shaped, so to assure the data flexibility to the detriment the data model. For example, Google IoT and IBM Watson use formats as JSON or XML.
4. **Verify the correctness of IoT Messages of IoT Devices.** The platform should be capable of verifying

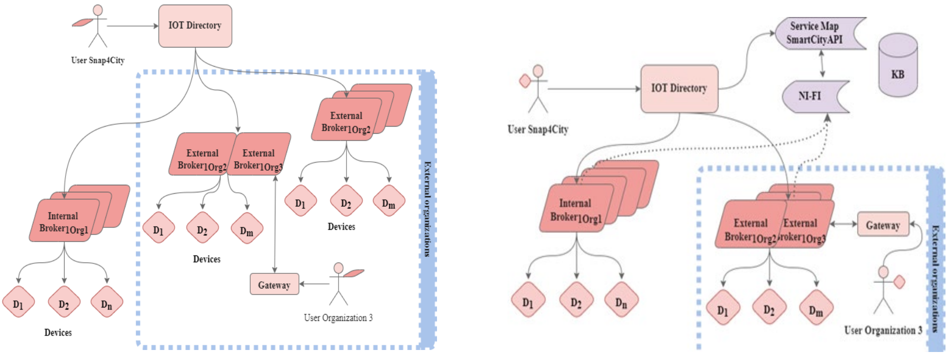


Figure 1. (left) registration of an IoT Broker, (right) registration of an IoT Device. The solid lines indicate the registrations, while the dashed lines indicate the data flow of the subscriptions.

correctness of messages in terms of model and format including verification at level of attributes, before accepting/sending them. Please note: this requirement is satisfied by each solution considered since the IoT Devices are formally defined at the registration phase.

5. **Semantic Interoperability.** This requirement is fundamental to achieve the coherence among different IoT Devices (e.g., provided by different builders, addressing the same concepts, information on attributes). An IoT Platform should be capable to recognize/classify/retrieve information/attributes and behave accordingly to the semantic data model and types. For example, an IoT application should not risk misunderstanding the unit of measure assigned to attributes of different devices which have the same name, but different units.
6. **Support automatics deploy of Internal IoT Brokers.** The IoT Platform should provide support for the automated deployment of IoT Broker internally managed. And thus, Internal Brokers are directly managed by the Platform which directly performs the registration of IoT Devices on them. The result is an easy experience for the user and an easy way to populate the network. This requirement can be implemented only if the Platform allows the registration and the management of new IoT Brokers. For this reason, this requirement is satisfied only in the Core of Siemens MindSphere and by all FIWARE Platforms for definition.
7. **Register External Brokers.** The platform must support the registration of IoT External Brokers. This means that the IoT Platform should be capable of registering IoT Devices/Services of the External Broker into the IoT Platform. Brokers can be single- or multi-tenant and to recover the IoT Devices data model managed by the Broker is the first step to perform their registration. In the case of External Broker, the endpoint URL and the service and/or service path specifications would be needed to subscribe. None of the commercial platforms considered provides a solution for registering External Brokers, and thus making automated registration of their devices.
8. **Discover IoT Devices on IoT Brokers.** The platform must be capable to abstract IoT Devices from their IoT Brokers and protocols. This is needed for their registration and for their **classification and search**, which is based on their position, nature, value types and units, etc. In other words, it should be possible to discover/search (subscribe, get, send data) to/from IoT Devices independently from their position/connection in the IoT Network. The process of discovery must be manageable in the sense that its execution time can be scheduled, and possible with brokers that support a process for device discovery. The result should consist of an automated or semi-automated registration process of IoT Devices.
9. **Easy management graphic interface to list and test IoT Brokers, and IoT Devices** and query them. For each IoT Device, it has to be possible to perform testing activities. As seen before, not all the above-mentioned Platforms manage the IoT Broker, unless they use a specific add-on. So, this requirement is satisfied by each of them only for the kind of Devices they put on the offer.
10. **Manage IoT Device Model and Device Data Type ownership and access grant**. This permits assignment/changing of the ownership and the creation of access grants to the entities (Brokers, Devices, Models, etc.). In delegation management, it must be possible to list them (check the grants provided) and revoke the delegations. According to GDPR, any entity must start as private of the owner. The delegation should be possible for organizations, groups of users, and single users.

On other aspects, surveys about IoT Platforms are provided in [14], [15], [16].

III. THE ROLE OF IOT DIRECTORY IN SNAP4CITY ARCHITECTURE

In order to enforce the above-described requirements into Snap4City IoT Platform, we have designed and developed a new concept that we have called IoT Directory. It extends the features of generic IoT Platforms with the management of (i) IoT Data Models, (ii) IoT External Brokers, (iii) discovery of IoT Devices of External Brokers, (iv) support the multi-tenancy, (v) support several organizations, (vi) GDPR compliance, etc.

In **Fig.1(left),** the registration process of IoT Brokers (**Internal or External**) in the Snap4City Platform is reported, where the organization is denoted by the subscription in the

Broker name. At the Broker registration into the IoT Directory, a number of parameters are needed including: the end point, security, name, External/Internal, single/ multiple-tenant, etc. The most important difference for Internal/external Brokers consists in the IoT Device management as explained in the following. The broker to be usable has to be granted to each specific user or public [13]. An user only belongs to a single organization for security and privacy aspects.

Once a Broker is registered, the IoT Directory automatically performs the subscription of the data platform to the new Broker for all its devices/topics, so that each new message that will be generated by the broker would be directly brokered to the data storage. In **Fig.1(right)**, the subscriptions are denoted by dashed lines, while the registrations are shown as solid lines. In most of the IoT Platforms, the storage is called Data Shadow, and allows to create the historical data of the IoT Devices. In Snap4City, the data storage feeding is performed by Apache NiFi, so that, the IoT Directory automatically performs the association between NiFi and the topics of the new **Internal or External** Broker. This is due to the necessity of having a robust, scalable tool with low latency to handle huge volume of data entering on the storage and coming from several devices and brokers. In **Fig.1(right)**, the processes of **IoT Device registrations** are depicted in both cases. In the case of **Internal Brokers,** the IoT Device registration is performed on the IoT Directory. The user may select the IoT Broker among those of the Organization and set a number of details. The registration may start with the exploitation of an IoT Device Model, the device ID, and then with the definition of GPS location, and all instance details. In Snap4City, the process can be performed: (i) on the IoT Directory via the user interface exploiting a model or not, (ii) exploiting API of the IoT Directory, (iii) using MicroServices in Node-RED which are based on the same API of (ii), (iv) using a set of automated registration processes starting from Excel Files/tables. Each registered IoT Device is registered on the **Knowledge Base**, KB, (implemented with Virtuoso on the basis of Km4City Ontology [17] for the semantic relationships and with Open Search for the time Series data) with all its information and metadata (static information). The KB management allows to index the device and establish all the relationships with the other city entities located in the same area, place, city, region, road, GPS position, etc. This information would be very useful when new messages arrive from a IoT Device in the storage via NiFi (which is represented by the ServiceMap in **Fig.1(right)**, with **Smart City API** for providing access to the storage) they are connected to the right IoT Device description and relationships. The correct and complete indexing and **Smart City API** are fundamental to enable the exploitation of IoT data by IoT Applications (Node-RED microservices [14]) and Dashboards [16].

Finally, **Fig.2** shows the data flow during the usage, it illustrates the event-driven data flows. There are four ways for generating new IoT Messages, from:

- **IoT Devices** pass from a Broker and are passed to: (a) the NiFi, thus reaching the KB and storage, becoming part of historical data which can be accessed and queried from IoT App, Data Analytic and Dashboards; (b) Kafka to directly reach subscribed Dashboards via WebSockets, and IoT App.
- **IoT Apps** may be sent to IoT Broker or to Kafka front end broker. Thus, the message can reach: IoT Device for acting on it, storage, IoT App, Dashboards, etc. If a message is sent by a sensor-actuator (Internal or External), his Broker broadcasts it to Ni-Fi, which spreads it in turn. The IoT Apps are also used for massive registration of IoT Devices, and to perform data adaptation, ETL/ELT (Extract Transform/Load) processes.
- **Dashboards** are passed via Kafka toward the IoT App, or to an Internal Broker or direly into the storage. These messages can be regarded as Virtual IoT Devices to act on some sensors/actuators IoT Device, or even to simulate it. The produce messages may be sent to Internal/External Brokers, and so on.
- **IoT Directory** may generate a new message towards an IoT Broker (and may also read the last message from the broker). The generation of messages from the IoT Directory is typically used to check if the Internal Broker is alive and works correctly, and if the IoT Device messages are accepted.

  In the case of **registration of IoT Devices on an External Broker**, the Broker is not managed by the IoT Directory, and thus the Devices are registered in the External Broker by the third-party gateway manager without informing the IoT Directory. Thus, the IoT Directory need to recover the information needed for registering and indexing the devices into the KB. Then, the IoT Directory queries/harvest the external broker to get the structures of the IoT Devices (also called Discovery has to be started). The harvesting process may start having the broker API end point and also the service path in the case of the multi-tenancy broker. The harvesting has to be performed at the broker registration and every time a new Device is added, thus a periodic Discovery is needed. For the harvesting, the IoT Directory recognizes the Data Model and Data Types of the attributes to register them in KB in proper manner, to validate them. In the following subsections, the registration of IoT Devices on Internal and External Brokers are discussed. Please note that the registration of devices from External Brokers is one of the innovative aspects addressed by the IoT Directory which is capable to harvest the brokers and resolving semantic gaps on IoT device attributes/variables, see **Section 3.2.**

A. Registration of IoT Devices on Internal Brokers

The IOT Directory is fundamental for the definition and registration of IoT Device Models, i.e., data models. Focusing on the Internal IOT Brokers, the platform provides three ways to deploy IoT Devices:

- *Manual process:* the user can register an IoT Device by using a graphic interface, to input information. It is possible to register a device on the basis of a specific IoT Device Model, and to refer at a specific IoT Broker.

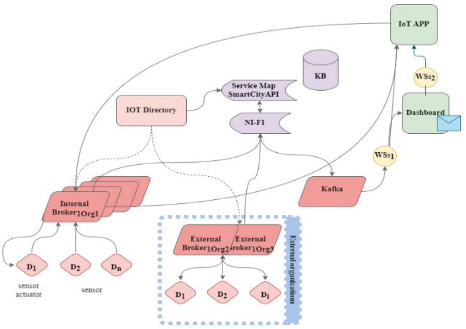


Figure 2. IoT Messages exchanged among entities: continuous lines are data flows, dashed lines indicate the tests that the user can perform to verify the IoT Devices/Brokers.

- *Bulk process:* the user can upload a file with a list of Devices, defining the IoT Broker, Model and Edge.
- *IoT App process:* The user can build an IoT APP using Node-RED on which specific nodes can be used. The nodes accept JSON with parameters related to the chosen Device Model to register new Devices (see **Fig.4**).

The registered IoT Devices are shown in a table in which the users can manipulate only those he/she created, no matter of the generation process. The users can also see in the list the public devices of the same organization, while the general administrator has a full visibility of all devices of all the organizations.

B. Discovering & Registering IoT Devices from External Brokers

The Snap4City Platform allows the registration of External Brokers using the broker URL or the couple URL and service identifier in the case of multi-tenancy. After the registration of an External Broker, the user can start the harvesting process and choose the time period for the update. It is important to highlight that Snap4City Platform may know a set of IoT Device Model (data models). Thus, in the best case, when the harvesting starts, it can recognize the device and its model (message format and device ID).

If the IoT Directory does not recognize the Device, the Device has to be Registered. To this end the IoT Directory can query the Broker to have more information to register it. On the other hand, the IoT Device may be compliant with a Data Model or not. If it is compliant the registration is straight forward. It is not compliant, each single attribute has to be recognized in terms of Value Type, Value Unit and Data Type (e.g., Temperature, Celsius, Float). The list of recognized/registered and non recognized/non registered Devices is presented. Through this interface, the user can resolve the problems manually defining the missing data and enabling the registration.

The most common harvesting (automated registration process) problem are due to the lack of matching with known attributes. The IoT Platform tries to identify the attribute kind in terms of value type, value unit and data type by performing query on data Dictionary and Km4City Ontology (via the connection from IoT Directory and ServiceMap by using Smart City API, and Dictionary API, **Fig.** 2. The recognition may have success in two cases: the data model is known but not this specific attribute or the model is not known, so all the attribute values of the Device are not recognized. In the first case is easy to fix the problem by applying a specific rule. In the second case, the Platform needs to learn a Rule for solving the attributes, thus defining a new Data Model in IoT Directory. Formally, the processing rule R is defined in EBNF as following:

R:= IF <condition list> THEN <action list>
<condition list>: = <c> | <c> AND <condition list>
<c> := <variable> <op> <constant>
<variable> := “device name” | “context broker” | “device type” | “model” | “value name”
<op> := “is equal” | “is not equal” | “Is null” | “contains”
<constant> := integer | float | string | list
<action list> := <a>| <a>, <action list>
<a> :=<action variable>: <action constant>
<action variable>:= “Data type” | “Value type” | “Value unit” | “Editable” | <Coded Healthiness criteria> | <Healthiness value>
<action constant>:= string

The rule is divided into two parts: If statement and then statement. In the first part, the user can define the condition that describes a set of devices, e.g., a device name in common. The *<op>* defines the operators, two of them (“is/ is not equal”) can apply only on the number constant, others (“Is null” or “contains”) only on the string constant. In the second part, the user can establish the action that makes devices or values valid. In other words, for the subset of Devices identify by the <condition list>, the *<action variable>* is modified as defined by *<action constant>*. An example of Rule can be:

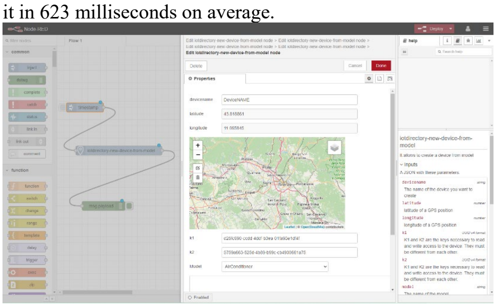
IF “context broker” is equal “CONTEXTBROKER”
AND “value name” is equal “activePower”
THEN “value type”:“power”, “value unit”:“W”, “data type”:“float”

In this case, the rule’s builder selects a subset of invalid attributes named activePower, which have the Device subscripts a Broker named CONTEXBROKER which allows to manage the multi-tenancy aspects. Then, it changes the value type, the value unit and the data type in power, W and float.

In the IoT Directory, it is possible to search and edit the saved rules. When the user saves a rule, must choose the broker to which is applied. It is also possible to define a specific subset of service or service and service path to cope with multiservice brokers. Thus, the application of a rule is associated to each specific Broker or Organization since a Rule can be suitable for an organization and not functional for the others. Figure 5 illustrates the form of the rule builder.

IV. VALIDATION EXPERIMENTS

As above discussed, the harvesting of External Brokers may take into account one or more rules to recognize the attributes and data models, and thus to indexing the IoT devices in the right manner, and shortening time to registration, dynamically add new IoT Devices registered on the Broker, thus reducing the gap from using Internal and External Brokers. The following validation values are calculated doing ad hoc experiments, which are repeated ten times. Focusing on the timing of the various processes, the user spends around 1 minute and a half on average to fill the form to add a new device with 10 attributes, and the system spends around 3 seconds to register the device. Meanwhile, if the user builds a device from a model, these timings are less than 1 minute on average to fill the form and around 1,35 seconds on average to submit the new device. Furthermore, if the user records a new device through IOT App, the system registers it in 623 milliseconds on average.



Edit iotdirectory-new-device-from-model node

devicename: DeviceNAME
latitude: 43.818861
longitude: 11.055845
k1: c25fc690-ccdd-4ddf-b3ea-01fa93e1df4f
k2: 5759a663-525d-4b89-b99c-cb4900661a75
Model: AirConditioner

Figure 3. User interface in Node-RED for the creation of the device by IoT App is shown.

Focusing on the registration timing and considering an external multi-tenant broker with 37000 devices, the harvesting time is 25 minutes and 50 seconds on average. Meanwhile, the process's timing of the attributes of a specific FIWARE data model (Streetlight) ingestion is 37.1406 milliseconds on average, which results in the addition of 432 new Streetlight devices automatically. Of course, the user can make changes to IoT Devices structure after their automated or manual registration.

V. CONCLUSIONS

The proliferation of the IOT devices, brokers, networks, data models, operators and tenant, make the harmonization and management for IoT Platform a hard goal. This paper offers an analysis and a comparison among relevant existing platforms and delineates the basics requirements to achieve these aims. These identified requirements are in most cases not addressed by major platform which prefer to stay on their own end-to-end solutions with limited interoperability and capacity of exploiting the legacy IoT networks in place. The interoperable management of complex network has to pass from the IoT device registration which is typically a recurrent operation since the IoT networks are in continuous evolution. In this paper, the above-mentioned problems have been addressed introducing our concept of IoT Directory and reasoning tools to (i) manage Internal and External brokers, (ii) perform the automated registration by harvesting and reasoning of devices managed into external brokers single- or multi-tenant services, (iii) perform the automated registration and management of Data Models, and any custom Data Model. The solution has been developed and tested into Snap4City, an 100% open source IoT platform for Smart Cities and Industry 4.0, official FIWARE platform, EOSC, and lib of Node-RED. Thus, the resulting platform is more flexible than the others considered (Google IOT Cloud, MS Azure, AWS, Siemens Mindshare and IBM Watson). Furthermore, the proposed solution is also compliant with Smart Data Model of FIWIRE. The semantic interoperability of the platform can be improved by automatic generation of rules and completing the automation of the ingestion process. Furthermore, the process is helped by Km4City ontology and Data Dictionary to recognize the new or model data's semantic domain. The specific IoT Directory has been developed in the context of Herit-Data Project, the results have been validated in wide condition of the whole Snap4City network of more than 18 tenant, and billions of data.

ACKNOWLEDGMENT

The authors would like to thank the MIUR, the University of Florence and the companies involved for co-founding the Herit-Data project. Km4City and Snap4City (https://www.snap4city.org) are open technologies and research of DISIT Lab. Sii-Mobility is grounded and has contributed to the Km4City open solutions.

REFERENCES

[1] Ghasempour, Alireza. "Internet of things in smart grid: Architecture, applications, services, key technologies, and challenges." Inventions 4.1 (2019): 22.

[2] Mekki, Kais, et al. "Overview of cellular LPWAN technologies for IoTdeployment: Sigfox, LoRaWAN, and NB-IoT." 2018 ieee international conferenceon pervasive computing and communications workshops (percom workshops).IEEE, 2018.

[3] Yousefpour, Ashkan, et al. "All one needs to know about fog computing andrelated edge computing paradigms: A complete survey." Journal of SystemsArchitecture 98 (2019): 289-330.

[4] Adelantado, Ferran, et al. "Understanding the limits of LoRaWAN." IEEECommunications magazine 55.9 (2017): 34-40.

[5] https://aws.amazon.com/iot

[6] https://siemens.mindsphere.io/en

[7] https://cloud.google.com/solutions/io

[8] https://azure.microsoft.com/en-us/overview/iot

[9] https://ww.ibm.com/watson

[10] https://www.snap4city.org

[11] Cheng B., G. Solmaz, et al.,"FogFlow: Easy Programming of IoT Services Over Cloud and Edges for SmartCities," in IEEE Internet of Things Journal, vol. 5, no. 2, pp. 696-707, April 2018,doi: 10.1109/JIOT.2017.2747214.

[12] Badii C., P. Bellini, A. Difino, P. Nesi, G. Pantaleo, M. Paolucci, "MicroServicesSuite for Smart City Applications", Sensors, MDPI, 2019. https://doi.org/10.3390/s19214798

[13] Badii C.:, P. Bellini, A. Difino, P. Nesi, "Smart City IoT Platform RespectingGDPR Privacy and Security Aspects", IEEE Access, 2020.10.1109/ACCESS.2020.2968741

[14] Ammar, Mahmoud, et all. "Internet of Things: Asurvey on the security of IoT frameworks." Journal of Information Security andApplications 38 (2018): 8-27.

[15] Ray, Partha Pratim. "A survey of IoT cloud platforms." Future Computing andInformatics Journal 1.1-2 (2016): 35-46.

[16] Bellini P., D. Cenni, et al, "Smart CityControl Room Dashboards: Big Data Infrastructure, from data to decisionsupport", Journal of Visual Languages and Computing, 10.18293/VLSS2018-030

[17] Bellini P., D. Nesi, et al, "Federation of Smart City Services viaAPIs", Proc of 6th IEEE International Workshop on Sensors and Smart Cities, with IEEE SmartComp, 14-17 Sept. 2020, Bologna, Italy. http://ssc2020.unime.it/

Addressing Data Security in IoT: Minimum Sample Size and Denoising Diffusion Models for Improved Malware Detection

Chiara Camerota*† Lorenzo Pappone† Tommaso Pecorella* Flavio Esposito†
* Department of Information Engineering (DINFO), University of Florence, Italy
†Department of Computer Science, Saint Louis University, USA

***Abstract*—Machine learning (ML) has emerged as a compelling approach to identify attacks in network traffic security. Existing malware detection strategies often concentrate on specific facets, such as efficient data collection, particular types of malware, or handling data scarcity. While valid, these strategies typically overlook the potential for minimizing sample size, focusing instead on data augmentation. This work introduces a novel method to determine the minimum sample size necessary to achieve a specified accuracy level, measured by the F1 score derived from the confusion matrix. We focus on TCP header traffic data transformed into images through flow-splitting techniques for multi-class traffic classification. In addition, we introduce a diffusion model to generate new synthetic traffic images and show that our method outperforms existing techniques in terms of stability and predictability. This study also compares the effectiveness of synthetic image augmentation using Generative Adversarial Networks (GANs) and Denoising Diffusion Probabilistic Models (DDPM) in improving image recognition and classification accuracy.**

***Index Terms*—malware detection, traffic classification, deep learning**

I. INTRODUCTION

Securing Internet of Things (IoT) applications is important but challenging for several reasons. One of them is the inherent limitations of low-power devices, which cannot adequately address robust protection strategies. Furthermore, consider the scenario in which an attack persists due to a lack of immediate identification: in such instances, the system may be severely compromised, potentially leading to significant financial losses for the network owner, as highlighted by [1]. This is particularly problematic because it compromises sensitive data and threatens physical and network security.

The application of machine learning (ML) has become increasingly prevalent in the detection of malware and other cyber threats [2], given its ability to process large data sets and decipher complex relationships between system variables [3]. Several ML approaches have been proposed to effectively detect previously seen and unseen malware, securing (IoT) networks [4].

While these approaches have merit, an accurate model often requires the availability of extensive and heterogeneous datasets, that are not always available or accessible. In addition, factors such as data scarcity and quality of training samples can affect the performance of classification models. As highlighted in [5], [6], the spurious correlation due to a lack of data and methodological rigor may lead to erroneous conclusions. Even when the ML model is rigorous, having a correct training dataset is the key. To address a robust model and limit spurious correlation, the literature investigates methods to determine the minimum sample size [5], [6]. In addition, in network security literature, a common solution in case of data scarcity is to implement a data augmentation strategy that enhances the learning model's robustness. Data augmentation involves artificially expanding a training dataset by generating modified versions of the original data without requiring new data collection [7]. Introducing noise and data variability, these techniques enhance the model's generalization ability and reduce overfitting [8]. A strategy to apply this technique is to feed the model with traffic transformed into images. In [9], for example, feature extraction from traffic-based-images is shown to be not only more efficient in terms of accuracy compared to the binary representation of the same packet but also allows the method scalability. As demonstrated e.g., in [10], a low-dimensional representation of the image simplifies its categorization.

Our Contribution. Motivated by these results, we show how new image generation techniques, such as diffusion model-based [11], combined with dataset management, can achieve higher accuracy and lower false positive rates in malware classification, compared to existing approaches. We propose a new approach to find a minimum sample size based only on the confusion matrix to address the lack of sufficient training. Moreover, we adopt Denoising Diffusion Probabilistic Models (DDPM) for the data augmentation and dissect its link with the false positive rate. *We found that the DDPM-generated data have a 7% higher F1 score and less variance (5%) than the Generative Adversarial Networks (GAN) generated data* and a higher average AUROC index along the classes. Also, through explainable AI techniques, we show how DDPM images are more similar to real with respect to those generated with GAN, hence validating our approach.

The rest of the paper is organized as follows. Section II outlines the state of the art, while section III highlights the problem definition and our contributions. In Section IV, we describe the experimental setup, detailing our methodology. Section V discusses the experimental results, providing insightful comments and interpretations. Finally, the paper concludes with a summary of the key findings and their implications.

II. RELATED WORK

Malware attacks pose an increasing threat to both IoT and traditional systems [12]. To mitigate IoT device damage, [13] aim to detect existing and new IoT malware by converting traffic data into RGB images. However, their feature pre-processing combines behavioral data with malicious activity frequency, which requires extensive data collection and may limit applicability in other environments. Similarly, in [14], Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTMs) based method is exploited to deal with malware traffic classification in high-variance daily energy consumption scenarios. Moreover, these work highlight the correlation between good prediction performance and the degree of diversity with datasets, in specific scenarios where the data are often unavailable. To the best of our knowledge, a method to achieve high accuracy with reduced data collection for IoT malware detection problems is still missing.

Data scarcity remains a significant challenge in many research domains, prompting the development of various mitigation techniques. Among these, data augmentation has emerged as a particularly effective and rapidly evolving method. Recent literature highlights diverse approaches to this problem, especially in specialized fields such as IoT security. [15] proposes a novel CNN architecture with dilated convolutions, channel squeezing, and boosting to identify IoT-specific malware patterns. While the study includes basic data augmentation methods like image rotation, these prove less effective due to the lack of natural directional orientation in the images. In contrast, [16] demonstrates a more sophisticated approach by integrating Generative Adversarial Networks (GANs) and Variational Autoencoders (VAEs) to improve malware traffic classification after generating synthetic malware samples for training. While their findings indicate that GANs could improve detection performance, GAN-generated traffic images still produce a high false negative rate. In fact, the synthetic images closely resemble the distribution of the original samples, and fail to introduce sufficient noise into the training set to achieve a better generalization.

To create realistic traffic-based images and identify traffic profiles, the author of [17] introduce a stable diffusion model for data augmentation. They demonstrated that profile detection improves by switching to a text-to-image generative model, requiring two datasets: one for RGB images and one for stable diffusion prompts. While this reveals the potential of denoising models in image generation, the process can be costly. This inspired the use of the DDPM as an image-to-image generative method, which offers high-fidelity images and more precise results in terms of Fréchet Inception Distance (FID) compared to GANs [18], [19].

III. PROBLEM DEFINITION: ADDRESSING DATA SCARCITY IN IOT MALWARE DETECTION

In traditional approaches (for IoT attack classification with or without complete or sufficient data), the accuracy of a model is directly linked to the amount of data it is exposed to. Researchers have shown that more data points allow the model to recognize a wider range of patterns and establish a more resilient understanding of the underlying relationships [20]. However, in the context of IoT malware detection, data are frequently lacking and costly to obtain, making it difficult to achieve accurate classification. Our aim is to address this challenge by collecting data over a short period, determined by the identified minimum sample size, and then classifying it over a longer timeframe. The objective is to categorize different types of malware when data collection is limited or expensive. To identify the minimum amount of data to collect, we propose a new method based on the confusion matrix, without distribution assumption, employed to obtain accurate results in terms of model accuracy and false/true positive rates. Data augmentation is also performed using the diffusion probabilistic model to create more stable and high-fidelity images.

IV. METHODOLOGY AND SOLUTION DESIGN

Data scarcity is a common issue in malware classification. In this work, we introduce a new method based on the confusion matrix and F-score to determine the minimum sample size for malware detection datasets. Our process does not need an assumption on the data distribution and identifies the minimum number of experiments for each class thanks to the McNemar-Bowker statistics test [21]. Furthermore, according to the data scarcity hypothesis, we explore the Denoising Diffusion Probabilistic Model to increase the data variability and make the classification model robust and precise [22].

A. Minimum Sample Size Definition

The literature explores the relationship between the number of examples considered (sample size) and model performance in ML. Studies such as [23] highlight that models with fewer parameters often perform well on smaller datasets. For instance, [24] demonstrates that simpler models can excel with limited data. Additionally, [25] investigated how the training set size impacts accuracy assessment, using a normal distribution assumption to define confidence intervals and calculate the sample size. However, this assumes linearity, which is not always empirically supported; in practice, the index curve often follows a more flexible distribution, like Beta, rather than a linear trend [26].

To address this issue, our method avoids assuming a specific distribution for the performance index. Instead, we use the confusion matrix (CF), from which performance metrics like the F1-score are derived. The CF is treated as the empirical joint probability distribution of the model's performance, and we perform statistical tests based on this matrix to quantify accuracy. Each CF element, excluding the diagonal, indicates the probability of misclassification—for example, a Distributed Denial of Service (DDoS) attack misclassified as a port scanning attack. The diagonal elements reflect the model's correct classifications. We apply the McNemar-Bowker test [21], a nonparametric test suited for comparing two related groups on a dichotomous variable. This test evaluates marginal homogeneity under the null hypothesis and follows a chi-square distribution with one degree of freedom.

Given the imbalanced nature of the malware dataset, the F1 score is an appropriate choice of performance index. We

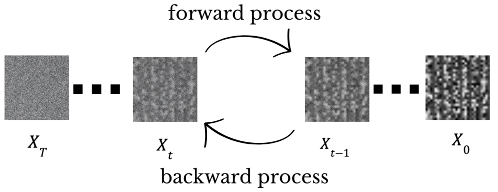


Fig. 1: The Denoising Diffusion Probabilistic Models' mechanism involves adding noise in the forward phase and learning noise removal techniques in the backward phase, without assuming data distribution

can define the F1 score in terms of true positive (TP), false negative (FN) and false positive (FP) values as follows:

$$F_1 \geq \frac{2 \cdot \mathrm{TP}}{2 \cdot \mathrm{TP} + 1 \cdot \mathrm{FN} + \mathrm{FP}} \,. \tag{1}$$

With this approach, we can consider the F1 score for each class and determine their sample size by applying the above definition to the test. Since a type I error (α) is a false positive conclusion in a statistic test and a Type II (β) error is a false negative conclusion, these terms are often used interchangeably with the general notion of false positives and false negatives mentioned in the formula.

B. Flow Image Generation

This subsection provides a concise overview of the generative models employed. In particular, the Deep Convolutional Generative Adversarial Network (DcGAN) and the denoising diffusion probabilistic model are considered.

A GAN architecture consists of a generator model that creates fake new samples as input to a discriminator model that determines if the input samples are fake or real. The generator learns to create traffic images as close as possible to real ones, while the discriminator checks their authenticity. The network aims to maximize the discriminator's loss (ensuring similarity between real and fake images) and minimize the generator's loss (improving its ability to mimic traffic behavior).

In our analysis, both losses are defined using Binary Cross Entropy. We apply the DcGAN model for its computational efficiency and ability to capture semantic features [27]. Here, the generator uses a one-dimensional convolutional network, reflecting the simplicity of the black-and-white data and minimizing computational cost.

The DDPM network takes two inputs, X_t (the final traffic-based image) and t (the steps that we want), and outputs a vector $\mu_\theta(X_t, t)$ and a fixed matrix $\Sigma_\theta(X_t, t)$, respectively the mean of the pixel and their variance matrix. This allows each step in the forward diffusion process to be approximately reversed by $X_{t-1} \sim N(\mu_\theta(X_t, t), \Sigma_\theta(X_t, t))$. The process is simplified and illustrated in Fig. 1. This method offers several advantages and addresses all the DcGAN issues listed above, including the ability to easily tune the model and generate more stable and performing outcomes in terms of image fidelity. In addition, the learning model to implement this kind of model can be chosen appropriately for the task.

In the literature, several neural networks are available, but for our purposes, lightweight models that capture key aspects of traffic images are most suitable. Therefore, we use a simple U-Net [28] with a 1-dimensional convolution layer. U-Net's ability to retain spatial information aids in recovering fine details and precise object localization. Its U-shaped architecture effectively combines local and global information, enhancing semantic accuracy compared to Convolutional Networks and common models like YOLO, which are slower and more resource-intensive. Additionally, U-Net reduces parameters, allowing faster training and lower computational demands. Its ability to learn from limited data and handle varying image sizes without preprocessing further adds to its practicality [28].

V. EVALUATION

The following sections explain how the application of a new method to define the sample size combined with data augmentation through the adoption of DDPM can achieve high results in terms of F1 score, taking into account the False positive rate. Also, a deep study of synthetic images using explainable AI techniques demonstrates how DDPM helps the classification model.

A. Experimental Setting

Figure 2 shows a summary of the experiment workflow. The packet capture (PCAP) files collected in *EDGE-IIOTSET* [29] and *Malicious Network Traffic PCAPs and binary visualization images Dataset* (MNT) [14] are considered and analyzed as images for this work. Specifically, the files are initially divided into unidirectional flows based on the same 5-tuple: source IP, source port, destination IP, destination port, and transport-level protocol. Subsequently, the TCP headers of each flow are concatenated. Additionally, they are truncated if they exceed 743 bytes; otherwise, zero padding is added, based on [30], where it is shown that the most pertinent information is concentrated in these initial bytes. The resulting files are converted to hexadecimal and then into 28x28 grayscale images. This choice is coherent with the data parsimony and offers a quick transformation that only needs the header information. The generative models, DcGAN and DDPM, are specific for each class considered and are trained on the sub-dataset to achieve superior prediction accuracy. Also, the final datasets include different proportions of synthetic data to explore the impact of synthetic data on resilience to data scarcity across methods. Considering our study and utilizing the formula 1, we can calculate the β value for the sample size. Let's suppose $F_1 = 0.8$ and $\alpha = 0.05$, we can calculate a β value of 0.128 ($\beta = 0.128$). Performing the McNemar-Bowker test, the sample size outcome is 735 examples for each class for EDGE-IIOTSET and 580 for the MNT; these differences are due to the different number of classes considered. After, a CNN with fixed parameters and hyperparameters was employed to investigate the influence of sample size and synthetic data on model performance. Utilizing the identical CNN, the baseline model was trained exclusively on the original data with a fixed sample size. All models were trained for 200 epochs with a

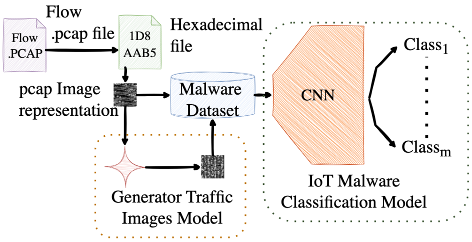


Fig. 2: The data flow starts with creating hexadecimal-based images for both datasets using the unidirectional flows header. After training the Generative model image-to-image for each class, the dataset is input to the CNN classification model to collect classification outcomes for evaluation.

TABLE I: F1 scores of the test set with a training model fed with the real dataset and the balanced dataset with a sample size less than the thresholds. The performance in other cases is worse than our method.

num. ex. per classes	**EDGE-IIOTSET**	**MNT**
$\leq$ threshold	0.6	0.7
real unbalanced train set	0.73	0.58
our train set	0.93	0.97

0.01 learning rate, employing stochastic gradient descent. Performance was evaluated with the F1 score on real, unbalanced test sets to evaluate the impact of dataset configuration on results.

B. Evaluation of Classification Performance

Figure 3 displays the confusion matrix for both baseline models using a Sankey diagram [31]. This diagram visualizes each class with boxes on either side, with green arrows indicating correct classifications and red arrows indicating misclassifications. The graph highlights dataset imbalances and critical classes. In the EDGE-IIOTSET, *uploading attacks* is the least accurately classified, while the *Java-RMI backdoor* is most frequently misclassified in the MNT set. Also, Table I shows that F1 scores degrade compared to our baseline when the number of examples per class falls below the threshold or when using the real unbalanced training set.

EDGE-IIOTSET. Focusing on the EDGE-IIOTSET, there are clear differences between the results obtained using GAN-generated images and DDPM-generated images, particularly regarding the classification of neutral traffic. It should be noted that DDPM consistently accurately tracks neutral traffic, regardless of the volume of synthetic data, thereby reducing misclassification variability. Conversely, models trained with GAN synthetic data exhibit increased misclassification, as shown in Figure 4. The plots highlight the challenge of accurately classifying minority classes, as shown by the curves. For instance, with 30%, 50%, and 60% synthetic data, the DDPM models struggle to classify uploading attacks correctly. However, in other cases, the models perform better across all critical classes. This may be due to increased noise from synthetic data and the limited number of training epochs set based on the baseline experiment. Moreover, Figure 5 displays the F1 score for each model along with the relative standard deviation. The baseline model, trained without synthetic data, exhibits high F1 scores and lower standard deviation values. Focusing on the other models, *we find that the DDPM models generally achieve better F1 scores but tend to have higher standard deviations,* except in the case where 30% synthetic data are used. This suggests that while DDPM models can enhance performance, they tend to exhibit greater variability, as said before. Another interesting aspect of these results is the DDPM-based model's capacity to classify better than the GAN-based one's class in the case of 10% synthetic data. However, the overall performance in terms of the F1 score suggests the opposite.

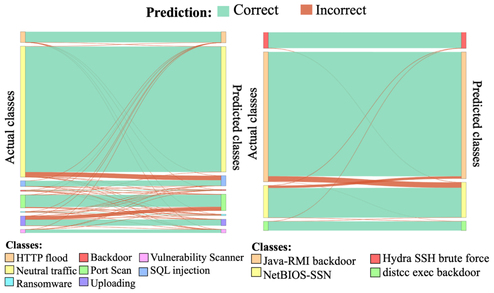


Fig. 3: The Sankey diagrams visualize the confusion matrix for EDGE-IIOTSET (right) and MNT set (left). The critical class to classify for the EDGE-IIOTSET is upload attacks, while for MNT, it's Java RMI backdoor attacks. These diagrams establish the baseline for classification and serve as a reference point for improvement using artificial images.

Malicious Network Traffic PCAPs Dataset. We further analyze the Malicious Network Traffic PCAPs and binary visualization images dataset. The main differences between this dataset and the previous one are the lack of neutral traffic and the smaller number of classes considered. However, the ROC curves shown in Figure 6 combined with the results in Figure 5 confirm the fact that DDPM-based models perform better than the GAN-based models, likewise for the critical class, both in terms of average F1-scores and their variance. The performance of both models declines significantly when the synthetic data percentage reaches 70%, suggesting that the models struggle with noise introduced by artificial datasets. This is particularly interesting given the previous ROC curve, where the same model struggled to discriminate classes with 30% synthetic data. Additionally, the GAN-based model trained with 90% artificial data failed. We extended the training by 200 epochs to validate these findings but observed no significant improvements, indicating that a more complex model may be necessary for this task. In conclusion, the DDPM-based synthetic images yield a more stable and predictable dataset compared to those based on GAN images. Additionally, the classification performance on the imbalanced test set is superior, as evidenced by reduced misclassification confusion, as shown in Figure 5.

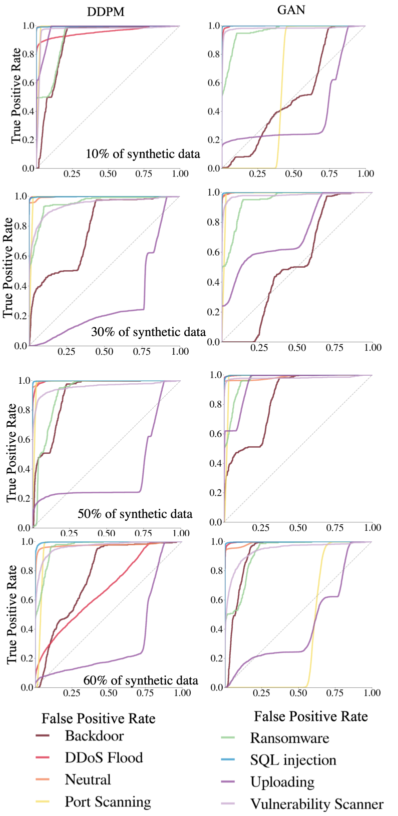


Fig. 4: ROC curve for EDGE-IIOTSET. These graphs plot the true positive rate against the false positive rate at each threshold setting. DDPM performs better than GAN, as the curve below the bisector line indicates. An increase in misclassification highlights the influence of synthetic data percentage on performance degradation.

C. *Pixels vs. Packets: Analysis of Synthetic Traffic via XAI*

Integrating synthetic traffic data into malware classification models introduces complexities that require a deeper understanding of model behavior. While performance metrics offer a wide measure of accuracy, they do not elucidate how synthetic data affects the classifier's decision-making process. In this context, explainable AI (XAI) is essential for validating the fidelity of synthetic traffic by comparing its impact on model decisions with that of real traffic data. This method helps evaluate synthetic data quality and reveals potential biases or artifacts introduced by various generation techniques. We use Gradient-weighted Class Activation Mapping (Grad-CAM) [32], a leading XAI technique, to analyze how synthetic traffic generation influences classifier decisions. This approach enables us to visualize and quantify the key features driving the CNN's predictions for both real and synthetic traffic flows.

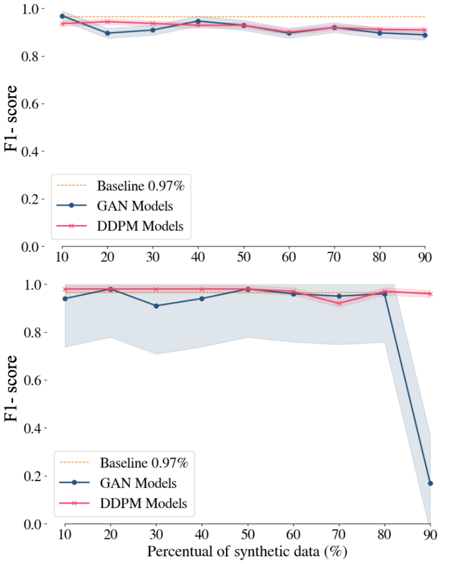


Fig. 5: Trend of class average F1 score for the test set with a *95% confidence interval*. The EDGE-IIOTSET plot shows that the DDPM-generated data have a 7% higher average F1 score and less variability than the GAN-generated data. The MNT F1 score trend curve highlights that DDPM-based models are more stable with minimal performance variance with more synthetic data. The performance of GAN deteriorates at 90% synthetic data.

Grad-CAM Heatmaps. We analyze the generative capabilities of both DDPM and GAN models by exploiting the Grad-CAM heatmaps. Figure 7 shows an example of the heatmaps generated by the Grad-CAM algorithm applied to a correctly classified flow sample of the SQL injection attack. To compare the generative capabilities of DDPM and GAN, we evaluated three CNN-based classifiers trained on different formulations of the EDGE-IIOT dataset: (i) real data only, (ii) DDPM-based data combined with real data, (iii) GAN-based data combined with real data. The heatmaps in Figures 7b, 7c, and 7d reveal that the CNN model considers various parts of the traffic flow header with differing levels of importance depending on the input dataset, suggesting the need for a deeper evaluation.

Fidelity Analysis. In this analysis, we aim to deepen the behavior of the classifier by observing how the synthetic traffic data impacts the classification process. In Figure 8, we observe notable differences between DDPM and DcGAN-generated synthetic traffic. Specifically, the classifier shows dependencies to different header fields for each traffic generation method. For DDPM-generated data, the TCP Acknowledge Number is crucial for most attacks, while no clear pattern emerges with DcGAN-generated data. Additionally, the classifier uses

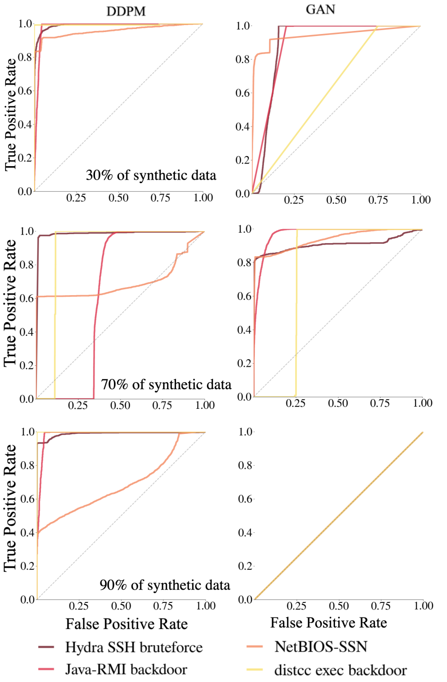


Fig. 6: ROC curve for the MNT Dataset. The DDPM's performances are better than the GAN's, and their curves appear below the bisector line. The DDPM can more effectively identify and classify different classes. When considering 90% of artificial data, the GAN-based model cannot achieve good classification, while the DDPM model can.

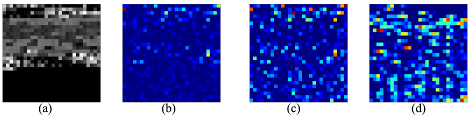
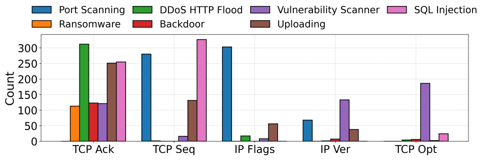


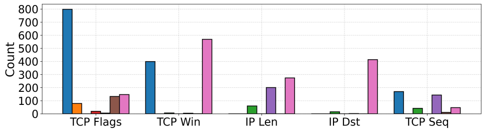
Fig. 7: Grad-CAM heatmaps of the SQL Injection attack. Red elements indicate a strong influence on the prediction, whereas blue elements have little effect on the prediction. (a) Original image of a SQL injection flow. (b) Resulting Grad-CAM of a CNN model trained only using real data (i.e., no synthetic traffic). (c) Resulting Grad-CAM of a CNN model trained with 50% traffic generated by DDPM. (d) Resulting Grad-CAM of a CNN model trained with 50% traffic generated by GAN.

distinct fields for each model: DDPM-generated traffic relies on a combination of TCP flags, TCP window size, IP header length, IP destination, and TCP sequence number to detect port scanning attacks, whereas GAN-generated traffic depends predominantly on TCP flags.

Packet Type Distribution. While header field analysis focuses on specific packet header values, we provide a higher-level view by examining the packet types associated with the most influential bytes using XAI techniques. We analyze TCP flags for each attack to assess classifier behavior with synthetic data from different generative models. Figures 9a, 9c show the distribution of TCP flags for correct classifications (Figures 9b, 9d) and misclassifications (Figures 9b, 9d). SYN packets are key for identifying attacks like backdoor and DDoS, appearing nearly 100% of the time in correct predictions. Exceptions include scanning attacks, where FIN+ACK and ACK packets are more relevant, and port scanning attacks, which feature high percentages of RST+ACK flags. SQL injection attacks show varied packets depending on the generative model. This analysis highlights that both generative models effectively capture attack behaviors, with SYN packets often linked to connection initiation attacks and other flags used for different

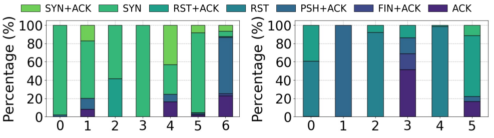


(a) DDPM.

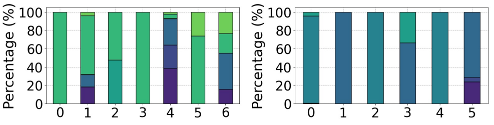


(b) DcGAN.

Fig. 8: (a) Distribution of top-5 header fields across attacks over DDPM-generated synthetic malware traffic. (b) Distribution of top-5 header fields across attacks over DcGAN-generated malware traffic.



(a) DDPM correct class. (b) DDPM misclassification.



(c) DcGAN correct class. (d) DcGAN misclassification.

Fig. 9: Distribution of TCP flags for packets determined as most influential in the classifier's predictions by the Grad-CAM algorithm. (a) TCP flag distribution for correctly classified DDPM-generated traffic. (b) TCP flag distribution for misclassified DDPM-generated traffic. (c) TCP flag distribution for correctly classified DcGAN-generated traffic. (d) TCP flag distribution for misclassified DcGAN-generated traffic.

types of attacks. Interestingly, Figures 9b and 9d reveal that the classifier mistakenly emphasizes RST packets for backdoor, port scanning, and SQL injection attacks. This suggests the classifier might be overfitting to synthetic data patterns that do not align with real-world attack behaviors. RST packets, often linked to rare abrupt terminations, may be overrepresented in synthetic data. To improve performance, future work should incorporate domain-specific knowledge of network protocols and attack patterns into both data generation and classifier training.

VI. CONCLUSION

In the IoT domain, data scarcity remains a significant challenge, often addressed by generating artificial data using DL-based generative techniques like GANs and their variants. This study investigates the efficacy of Denoising Diffusion Probabilistic Models (DDPM) and Generative Adversarial Networks (GAN) in augmenting limited training data for large-scale malware traffic image classification. Our findings indicate that DDPM consistently outperforms GAN as the proportion of synthetic data increases. We propose a novel method to determine the minimum sample size needed to achieve the desired F1-score accuracy. Future research directions include validating our methodology across diverse data sources and IoT contexts, as well as refining DDPM to enhance its specificity for IoT-related data generation.

ACKNOWLEDGEMENT

This work has been partially supported by the National Science Foundation (NSF) awards # 2133407 and # 2201536. Also, it was partially supported by the European Union under the Italian National Recovery and Resilience Plan (NRRP) of NextGenerationEU, a partnership on "Telecommunications of the Future" (PE0000001 - program "RESTART").

REFERENCES

[1] V. Hassija, V. Chamola, V. Saxena, D. Jain, P. Goyal, and B. Sikdar, "A survey on iot security: application areas, security threats, and solution architectures," *IEEe Access*, vol. 7, pp. 82 721–82 743, 2019.

[2] A. Souri and R. Hosseini, "A state-of-the-art survey of malware detection approaches using data mining techniques," *Human-centric Computing and Information Sciences*, vol. 8, no. 1, pp. 1–22, 2018.

[3] J. Sevilla, L. Heim, A. Ho, T. Besiroglu, M. Hobbhahn, and P. Villalobos, "Compute trends across three eras of machine learning," in *2022 International Joint Conference on Neural Networks (IJCNN)*. IEEE, Jul. 2022.

[4] K. Shaukat, S. Luo, and V. Varadharajan, "A novel deep learning-based approach for malware detection," *Engineering Applications of Artificial Intelligence*, vol. 122, p. 106030, 2023.

[5] W. Ye, G. Zheng, X. Cao, Y. Ma, X. Hu, and A. Zhang, "Spurious correlations in machine learning: A survey," *arXiv preprint arXiv:2402.12715*, 2024.

[6] A. L'heureux, K. Grolinger, H. F. Elyamany, and M. A. Capretz, "Machine learning with big data: Challenges and approaches," *Ieee Access*, vol. 5, pp. 7776–7797, 2017.

[7] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of big data*, vol. 6, no. 1, pp. 1–48, 2019.

[8] D. A. Van Dyk and X.-L. Meng, "The art of data augmentation," *Journal of Computational and Graphical Statistics*, vol. 10, no. 1, pp. 1–50, 2001.

[9] L. Nataraj, V. Yegneswaran, P. Porras, and J. Zhang, "A comparative assessment of malware classification using binary texture analysis and dynamic analysis," in *Proceedings of the 4th ACM Workshop on Security and Artificial Intelligence*, ser. AISec '11. New York, NY, USA: Association for Computing Machinery, 2011, p. 21–30.

[10] A. Oliva and A. Torralba, "Modeling the shape of the scene: A holistic representation of the spatial envelope," *International journal of computer vision*, vol. 42, pp. 145–175, 2001.

[11] H. Chen and M. A. Babar, "Security for machine learning-based software systems: A survey of threats, practices, and challenges," *ACM Computing Surveys*, vol. 56, no. 6, pp. 1–38, 2024.

[12] Ö. A. Aslan and R. Samet, "A comprehensive review on malware detection approaches," *IEEE access*, vol. 8, pp. 6249–6271, 2020.

[13] J. Jeon, J. H. Park, and Y.-S. Jeong, "Dynamic analysis for iot malware detection with convolution neural network model," *IEEE Access*, vol. 8, pp. 96 899–96 911, 2020.

[14] R. Chaganti, V. Ravi, and T. D. Pham, "A multi-view feature fusion approach for effective malware classification using deep learning," *Journal of information security and applications*, vol. 72, p. 103402, 2023.

[15] M. Asam, S. H. Khan, A. Akbar, S. Bibi, T. Jamal, A. Khan, U. Ghafoor, and M. R. Bhutta, "Iot malware detection architecture using a novel channel boosted and squeezed cnn," *Scientific Reports*, vol. 12, no. 1, p. 15498, 2022.

[16] M. Abdelaty, S. Scott-Hayward, R. Doriguzzi-Corin, and D. Siracusa, "Gadot: Gan-based adversarial training for robust ddos attack detection," in *2021 IEEE Conference on Communications and Network Security (CNS)*. IEEE, 2021, pp. 119–127.

[17] X. Jiang, S. Liu, A. Gember-Jacobson, A. N. Bhagoji, P. Schmitt, F. Bronzino, and N. Feamster, "Netdiffusion: Network data augmentation through protocol-constrained traffic generation," *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, vol. 8, no. 1, feb 2024.

[18] X. Su, J. Song, C. Meng, and S. Ermon, "Dual diffusion implicit bridges for image-to-image translation," *arXiv preprint arXiv:2203.08382*, 2022.

[19] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *Advances in neural information processing systems*, vol. 33, pp. 6840–6851, 2020.

[20] X. Ying, "An overview of overfitting and its solutions," in *Journal of physics: Conference series*, vol. 1168. IOP Publishing, 2019, p. 022022.

[21] M. Eliasziw and A. Donner, "Application of the mcnemar test to non-independent matched pair data," *Statistics in medicine*, vol. 10, no. 12, pp. 1981–1991, 1991.

[22] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," 2020. [Online]. Available: https://arxiv.org/abs/2006.11239

[23] C. Andrade, "Sample size and its importance in research," *Indian journal of psychological medicine*, vol. 42, no. 1, pp. 102–103, 2020.

[24] A. Gosiewska, A. Kozak, and P. Biecek, "Simpler is better: Lifting interpretability-performance trade-off via automated feature engineering," *Decision Support Systems*, vol. 150, p. 113556, 2021.

[25] G. M. Foody, "Sample size determination for image classification accuracy assessment and comparison," *International Journal of Remote Sensing*, vol. 30, no. 20, pp. 5273–5291, 2009.

[26] K. H. Brodersen, C. S. Ong, K. E. Stephan, and J. M. Buhmann, "The balanced accuracy and its posterior distribution," in *2010 20th International Conference on Pattern Recognition*, 2010, pp. 3121–3124.

[27] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *Advances in neural information processing systems*, vol. 27, 2014.

[28] N. Siddique, S. Paheding, C. P. Elkin, and V. Devabhaktuni, "U-net and its variants for medical image segmentation: A review of theory and applications," *IEEE access*, vol. 9, pp. 82 031–82 057, 2021.

[29] M. A. Ferrag, O. Friha, D. Hamouda, L. Maglaras, and H. Janicke, "Edge-iiotset: A new comprehensive realistic cyber security dataset of iot and iiot applications for centralized and federated learning," *IEEE Access*, vol. 10, pp. 40 281–40 306, 2022.

[30] *USTC-TK2016 Repository*. [Online]. Available: https://github.com/yungshenglu/USTC-TK2016/tree/ubuntu

[31] P. Riehmann, M. Hanfler, and B. Froehlich, "Interactive sankey diagrams," in *IEEE Symposium on Information Visualization, 2005. INFOVIS 2005*. IEEE, 2005, pp. 233–240.

[32] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 618–626.

The intrinsic convenience of federated learning in malware IoT detection

Chiara Camerota
DINFO
Università degli Studi di Firenze
Florence, ITALY
chiara.camerota@unifi.it

Tommaso Pecorella
DINFO
Università degli Studi di Firenze
Florence, ITALY
tommaso.pecorella@unifi.it

Andrew D. Bagdanov
DINFO
Università degli Studi di Firenze
Florence, ITALY
andrew.bagdanov@unifi.it

***Abstract*—The Internet of Things is emerging as a key concept, defining a network of interconnected devices capable of seamless data collection, exchange, and analysis. However, due to their emphasis on simplicity, these devices are often vulnerable to malware attacks. This study examines the potential of machine learning methods, specifically in the context of Federated Learning, to enhance privacy protection and to benefit from IoT's decentralized nature, such as the low overhead traffic. The proposed approach is a federated machine learning algorithm based on a central aggregator and several clients. The study aims to conduct a comprehensive analysis using the IOT-23 dataset, which contains real and labeled instances of malware infections. The test outcomes demonstrate that the proposed approach outperforms centralized approaches regarding the global area under the precision-recall curve (AUPRC) and variance, with a significance level of 0.05.**

***Index Terms*—AI - ML techniques, anomaly detection, federated learning, Internet of Things**

I. INTRODUCTION

The Internet of Things (IoT) refers to a network of interconnected devices and objects that can collect, exchange, and analyze data, allowing them to communicate and interact seamlessly [1]. These devices feature lighter protocols, lower power consumption, and compact shapes, allowing flexibility and adaptability [2]. In addition, this kind of Internet can be implemented in several wireless networks in order to become adaptable in various use cases; for example, a LoRaWAN connection can be chosen in smart cities, while for smart homes, the Z-Wave strategy is more appropriate. Likewise, smart devices have a wide range of applications, many in sensitive areas. However, no standard supports all smart devices, and built-in security mechanisms are not yet standardized. Furthermore, no proven security methods to guarantee the digital security of these infrastructures [3]. In addition, IoT devices prioritize simple operation over robust security measures, making them vulnerable to malicious users who can coordinate attacks through malware designed to cause damage, disrupt, or gain unauthorized access to a system [4]. Therefore, it is necessary to implement an intelligent, light, and flexible solution that can learn various attack patterns and does not interfere with the network's bandwidth or usability.

Furthermore, these methods must be able to forget learned patterns and re-learn when malware evolves [5]. In literature, deterministic techniques are commonly used in conjunction or not with Machine Learning (ML) methods, depending on the main aspect the tool should preserve. A key work that helps us understand this concept is offered by [6], which discusses anomaly detection analyses for different phases of the malware life cycle. Statistical analysis is utilized during the pre-execution phase, dynamic analysis is employed during the execution step, and memory analysis is conducted during the post-execution step. Conversely, malware detection is often considered a stand-alone problem and does not consider the network's topology or distributed nature. In [7], the authors take into account specific IoT challenges, such as privacy, but do not consider the impact of the model on the wireless network. Deeper, centralized models often are not appropriate for IoT cases because they can be accurate but may make the communication channel busy due to data exchange [8].

Federated learning (FL) is a suitable strategy to minimize data exchange. Furthermore, it is a distributed machine learning algorithm that benefits from the topology of IoT networks. FL employs an iterative approach with discrete interactions between the client and server, known as federated learning rounds, to achieve results comparable to those of traditional machine learning models. Therefore, this method has the potential to access a large amount of data while maintaining privacy guarantees.

This paper addresses the critical challenge of malware detection within the Internet of Things paradigm, focusing on the pivotal role of advanced models such as Federated Learning. We explore the adaptability of these models to the unique constraints of IoT environments and thoroughly examine the open challenges associated with deploying machine learning models in real-world scenarios where resources are finite and operational constraints are stringent. Our approach centers on developing a federated machine learning classifier as a practical and deployable solution for identifying malicious traffic within IoT networks. Using data generated from traffic packet analyzers, we outline effective strategies for data definition, preprocessing, and the deployment of ML algorithms. This

---

This work was partially supported by the European Union under the Italian National Recovery and Resilience Plan (NRRP) of NextGenerationEU, partnership on "Telecommunications of the Future" (PE0000001 - program "RESTART".

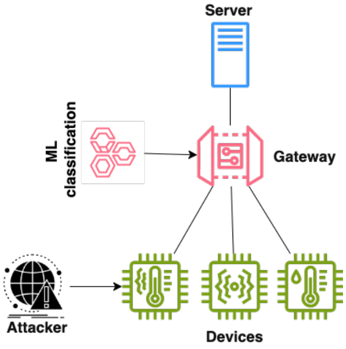


Fig. 1. A simplified representation of IoT system. The devices are low-powered, so the ML classifier is installed on the gateway to enhance the environment's security.

results in a robust model characterized by low variance in accuracy, making it suitable for resource-constrained environments. Additionally, we highlight the advantages of FL in reducing information leakage from devices and enhancing network performance. Our findings underscore the importance of advanced ML techniques in strengthening IoT ecosystems against evolving cyber threats, ensuring the integrity of interconnected devices. For further clarification, please refer to the diagram in Fig. 1. This illustrates the deployment of the ML classifier on the gateway and the execution of the attack on the devices. It should first be noted that the server is not being considered a point of interest in this context. The paper is structured as follows. Section II examines the current state-of-the-art in malware detection, which is the basis for the proposed model and methodology, such as the definition of problems explained in III. Section IV is dedicated to the proposed model and methods, in which the primary aspect and novelty of the proposed model are discussed. The next section discusses the results and their implications. After that, we present the future challenges and the conclusions.

II. STATE OF ART

The field of malware detection is extensive and encompasses numerous case studies. Malware attacks increasingly threaten both the IoT and traditional systems [9].

The literature contains several strategies for addressing this issue, which are collected, explored, and classified due to the heterogeneity of the problem and the available solutions. Machine learning techniques can help to analyze dynamics and memory by generalizing tasks with relevant data. Nascita's study shows that using customized data for IoT to feed an ML model improves performance [10]. This is also supported by [11], who demonstrate how to mitigate damage to IoT devices by intelligently identifying both known and emerging IoT malware. They achieve this by analyzing several device properties, transforming them into images, and applying dynamic analysis. However, these studies often overlook crucial factors such as the high cost of collecting extensive datasets, the scarcity of available data, and, most importantly, the overhead traffic generated by these approaches.

Another important aspect of machine learning models is their ability to be integrated into a cloud environment. However, this presents a complex challenge. A decentralized model should be considered over a centralized model to mitigate data exchange and communication channel availability issues. For this reason, Federated Learning techniques address this challenge and ensure privacy. They enable machine learning model training on the device and exchange only parameters with the server without sharing the user's information.

An example of FL applied to an IoT network is [12]. The authors present a model where the federated approach prioritizes participant privacy while achieving results comparable to those of centralized models. The authors also investigate the safety and robustness of this approach and demonstrate that the baseline model aggregation averaging step is highly vulnerable to attacks, even with a single adversary. However, they do not consider the impact on the network or analyze the comparison regarding the results' variance.

Another reason to consider decentralized methods over centralized ones is the challenge of combining anonymized data from heterogeneous and differently shaped data sources. Anonymized data alone cannot ensure equal client distribution, and a centralized model may face difficulties in handling the non-independent and identically distributed (non-IID) nature of device data. As a result, this can lead to potential bias and reduced accuracy during model training. In this situation, clients may be unreliable and experience higher failure rates or dropouts due to their dependence on less robust communication media, such as Wi-Fi, and power-constrained systems, such as smartphones and IoT devices. The authors of [13] suggest a novel federated learning approach to address privacy, robustness, and model training, concerns when dealing with non-IID data. Still, they focus our work exclusively on the ML model aspects and do not consider the environment. For interested readers, [14] comprehensively analyzes various FL methodologies that apply to IoT systems. Additionally, reference [15] outlines potential research topics.

III. PROBLEM DEFINITION: HOW CAN WE MINIMIZE THE VARIABILITY IN THE RESULTS?

In the Internet of Things networks, achieving efficient malware detection while balancing key factors such as privacy, low computational overhead, and effective data exchange presents a complex challenge. The objective is to develop a robust malware detection method that provides high accuracy and maintains stability over time, with minimal variation in accuracy across different scenarios. This involves a state-of-the-art analysis to identify an optimal detection model and deploy it in a manner that balances the critical points mentioned above. The investigation analyzes a subset of

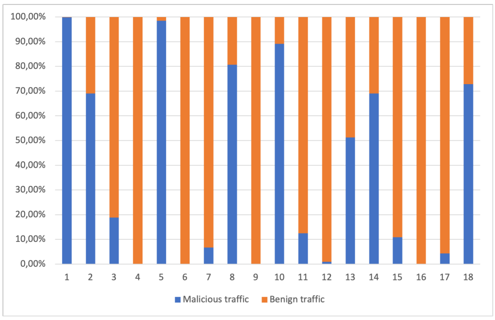


Fig. 2. Distribution of Malicious and Benign across the clients (horizontal axis).

overhead features from the IoT-23 dataset developed by Avast Software in Prague. The label distribution across the dataset is illustrated in Fig. 2, while a deep description is presented in the appendix. The challenges encountered included varying data set shapes, a significantly unbalanced label distribution, and a limited number of correlated features. A detailed data pre-processing investigation was also conducted. The objective of the research is to transform the limitations of IoT networks into strengths within a generalized model. Distributed approaches enhance scalability, whereas centralized models that exchange substantial data volumes between servers and clients can strain bandwidth. In federated models, only model parameters are shared with all clients, eliminating the necessity for each clients to transmit their own data and the relative parameters, thus reducing bandwidth usage and enhancing efficiency [16].

IV. METHODOLOGY AND SOLUTION PROPOSED

In this section, we present the pre-processing data methods and models. We briefly explain the standardization methods and the federated models used in the classical and modified versions. In conclusion, the evaluation methods are presented to describe the analysis in detail.

A. *Pre-processing data approach*

The pre-processing methodology became crucial in this work due to the variety of data and the high correlation between the features. For this reason, we introduce the core ideas considered and their implications. The following paragraphs present local and global standardization and distributed principal component analysis. These techniques address the high variance present in the data and make them homogeneous to simplify classification.

a) Local and global standardization: Standardization is a statistical technique that reshapes the data to make them more homogeneous and easy to classify. Given a client s with i examples in the dataset D_s, the local standardized data z_{si} is a transformation of the original data $d_{si} \in D_s$ as follows:

$$z_{si} = \frac{d_{si} - \mu_s}{\sigma_s} \tag{1}$$

where μ_s is the average vector of D_s and σ_s is the vector of standard deviation based on D_s. Otherwise, the global standardization replaces the μ_s and σ_s arrays with the μ and σ vectors, which are the mean and the standard deviation arrays calculated across all clients. These approaches offer valuable techniques for standardizing data, minimizing shared information, and enhancing model robustness.

The local standardization approach improves client confidentiality by modifying data within each client's domain. This method avoids transmitting sensitive information, such as the client's mean vector and variance, to the central server. Normalizing data locally helps address non-identically and independently distributed (non-IID) data across the network, making input data more uniform across clients. In contrast, global standardization involves estimating the overall data distribution on the server side. During the initial round of Federated Learning, the server aggregates mean and variance vectors from all participating clients, providing a comprehensive view of the network's data distribution. This global insight enables more robust model training by addressing disparities in data across clients. Both approaches offer distinct benefits: local standardization prioritizes privacy, while global standardization focuses on improving model performance by leveraging more comprehensive data distribution information. The choice between these methods depends on the federated learning application's specific needs and constraints.

b) Principal Component Analysis: The distributed version of Principal Component Analysis (PCA) [17] is presented. A classical PCA aims to find a smaller dimensional subspace that captures as much of the variance of the data as possible. The method involves a collaborative approach where the variance matrix is constructed using data from multiple clients or nodes in a distributed system. In this framework, each client processes and summarizes its local data, which is then exchanged with a central server or coordinator responsible for aggregating this information into a global variance matrix. Efficient computation of principal components necessitates the exchange of summarized indices or statistical measures between clients and the server. Nevertheless, this communication process may result in overhead due to data transmission and coordination. Despite the potential trade-off in increased communication complexity, the advantages of distributed PCA are consequential.

One advantage of PCA is its enhanced scalability to larger datasets that exceed the capacity of a single computing node or device. Distributing the computational load across multiple nodes reduces the overall processing time, leading to faster computation and analysis. Additionally, the distributed PCA model provides improved fault tolerance and resilience. Distributing data and computations across multiple nodes makes the system less susceptible to failures or disruptions in individual components. This redundancy can enhance the reliability and robustness of PCA computations in large-scale distributed environments. Furthermore, these distributed approaches also enable data analysis that preserves privacy. Clients can maintain control over their raw data while contributing only aggregated statistics or transformed representations to the central server. These strategies demonstrate proactive steps toward effective information management and collaborative model development within the context of FL. Each approach tackles unique challenges posed by distributed data settings, ultimately contributing to the advancement of secure and efficient machine learning practices.

B. Models

This section outlines our adopted models, introduces the base model, and describes our modifications. Specifically, the federated models discussed here are adaptations of Federated Averaging (FeAvg) and Federated Knowledge Distillation (FD). For a more detailed explanation of these models, please refer to [18], [19] and [20].

a) Classical version: The federated models utilize an iterative approach that entails discrete interactions between clients, who possess a local model, and the server, which establishes the global model. The server chooses a subset of clients to train the local model in each iteration and updates the global model based on the parameters received. The logit function is the main distinguishing factor between the two approaches.

For the FedAvg algorithm is shown in Fig. 3, while the loss function of a generic s device is given by:

$$L_s(D_s, \mathbf{w}) = \frac{1}{N_s}\sum_{i=1}^{N_s} l(d_{si}; \mathbf{w}) \tag{2}$$

where D_s corresponds to the dataset of device s, N_s corresponds to the amount of examples list in D_s, l is an entropy function such as cross-entropy function and d_{si} is the i-th example of D_s. It is important to note that the weight $\boldsymbol{w}$ is usually a simple averaging of the selected clients' weights.

On the other hand, Federated Knowledge Distillation involves a more complex loss function and requires selected clients to store the mean logit vector per label during local training. A simplification of the algorithm is shown in Fig. 4. The logit is a combination of the previous logit and the distance between the client and server logit arrays, as defined by the formula:

$$L_{KD}(D_s, \mathbf{w}) = L_s(D_s, \mathbf{w}) + \lambda \cdot KL_{dist,y}(Y_{log}, Y_{log,s}) \tag{3}$$

where $L_s(D_s, \mathbf{w})$ was defined previously and $KL_{dist,y}(Y_{log}, Y_{log,s})$ is the Kullback-Leibler metrics distance used to compare the logit distribution of the client and the server. The pre-trained phase, typical of this method, was excluded from the experiment to create a lightweight and fast tool.

b) Our modified version: The difference from the usual methods is that the update of the local weights of device s at step $t+1$ is calculated using the following formula:

$$\mathbf{w}_{s,t+1} = \beta \cdot \mathbf{w}_{s,t-1} + (1-\beta) \cdot \mathbf{w}_t \tag{4}$$

where $\mathbf{w}_{s,t-1}$ is the weights vector of the device s at step $t-1$, and $\mathbf{w}_t$ is the average weights vector of the selected device at step t. β is a hyperparameter that can take values from 0 to 1. In our experiment, we used $\beta = 0.75$. This approach guarantees stable weights and robust model performance, enabling the model to learn effectively without being influenced by variations in scale, such as alternating between high-value and low-value clients across training rounds.

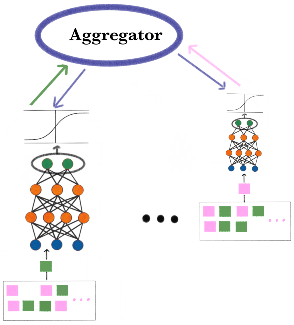


Fig. 3. Representation of a Federated Average method. The blue arrows communicate the average weights vector, while the other arrows communicate the client weights vectors. The boxes indicate the devices.

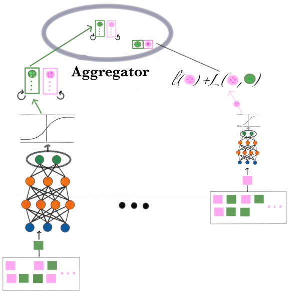


Fig. 4. Representation of a Federated Distillation method. The circle represents the different logit vectors: the full-colored ones are the global logit, the circle with the x in the center is the contribution for the regularization, and the others are the client logits. The boxes indicate the devices.

Our methods assume that the system is stationary, meaning that the statistical properties of the data, such as the mean and variance, remain constant over time. In a stationary system, the data distribution does not change significantly, allowing the model to rely on consistent patterns. However, these estimates may become inaccurate if the global mean and variance shift over time. To address potential non-stationarity, we trigger an index update when there is a significant change in average loss. This mechanism helps maintain the accuracy of the estimates and allows for dynamic adjustments. Specifically, the update is initiated when the loss change exceeds a threshold value of 0.1 (*threshold* > 0.1). As previously highlighted, minimizing information exchange and efficiently preprocessing data are crucial for enabling rapid analysis and easy deployment in practical scenarios.

C. Evaluation approach

To facilitate the comparison of results, the area under the precision-recall curve (AUPRC) is chosen as an index of fitness. The AUPRC is a widely recognized machine learning metric for evaluating classification algorithms. Consider the unbalanced nature of the data and provide a suitable summary of the fitness of the model on the data [21]. This index helps to identify the balance between precision and recall at various thresholds. Precision is the ratio of correctly classified examples to all positively classified examples, while recall is the ratio of correctly classified positive examples to all positive examples.

When comparing different models, their AUPRC scores will be compared by ratio. For instance, let's consider a generic Federated Model (AUPRC(Fed)) and a Centralized one (AUPRC(Centr)), the ratio of their AUPRC scores are calculated as follows:

$$\mathrm{AUPRC}_{\mathrm{Ratio}} = \frac{\mathrm{AUPRC(Fed)}}{\mathrm{AUPRC(Centr)}} \tag{5}$$

A ratio greater than 1 indicates that the Federated Model outperforms the Centralized one regarding AUPRC, while a ratio less than 1 indicates the opposite.

In conclusion, a Chi-square test can be performed to verify the variance reduction and evaluate whether the variances of two populations are equal. By applying this test, we can establish whether the variance reduction is meaningful and quantify the effectiveness of the proposed methods. This evaluation offers robust metrics that assure improvement.

V. EVALUATION

Our experiments want to highlight how the federated models overcome the centralized model, which is used as a baseline. That happens because the proposed model is more suitable for the nature of the IoT environment and allows some key concepts. Also, the exchange data can be manipulated depending on bandwidth availability.

A summary of the results can be found in the box plots presented in Figure 5. These box plots illustrate the distributions of the AUPRC values for each data configuration and operation method. The PCA and standardized transformation results were aggregated for the centralized approach as they produced identical outcomes. In contrast, the federated approach without standardized data led to the presentation of both models' outcomes. Upon examination of the figure, it becomes evident that the federated model with globally standardized or PCA data exhibits minimal variance and demonstrates superior global AUPRC performance. A chi-square test is performed to verify this reduction in variance, and the results are reported in Table I. The test results demonstrate that the Federated Learning approach significantly reduces variance in the Distributed PCA AUPR index and the Global Standardized AUPR index compared to the local Standardized Data Index, with a significance level of 0.05.

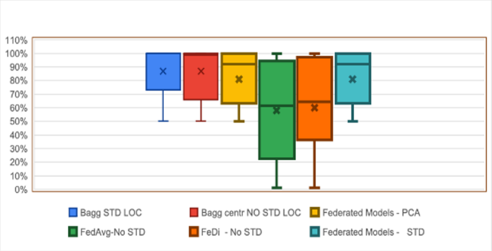


Fig. 5. Box plots of the AUPRC of the clients. The cross in each box represents the mean. As we can see, the low variance is achieved by the Federated models.

TABLE I
SUMMARY OF THE AVERAGE AUPRC RATIO BETWEEN FEDERATED MODELS (FEDAVG AND FD) AND CENTRALIZED MODELS

Model	Federated No STD	Federated Global STD	Federated PCA
Centralised No STD	0.94 (FedAvg) 1.07 (FD)	1.64	1.65
Centralised Data Transformation	0.82 (FedAvg) 0.97 (FD)	4.9	4.91

As mentioned above, analyzing model performances based on their AUPRC ratios provides valuable insight into different configurations. In the case of using only transformed data, FL methods exhibit significantly higher performance, approximately three times on average (with $\mathrm{AUPRC}_{Ratio} \simeq 4.9$), whether employing PCA-transformed data or standardized data. This outcome illustrates these methods' efficacy in reducing the variance across the results obtained. It can be seen that FedAvg demonstrates superior performance in this regard, as it minimizes the information exchange between nodes. In comparison to a centralized model, excluding standardized data during federated learning resulted in an average performance improvement of approximately 7% ($\mathrm{AUPRC}_{Ratio} = 1.07$). Despite this improvement, the significant variance observed suggests that while this federated model shows promise, it may not offer the most consistent results.

Further comparison between standardized data and transformed data within the FL framework revealed a substantial performance gain, with an average improvement of around 60% compared to traditional centralized methods

TABLE II
RESULT OF CHI TEST ON AUPRC INDEX PERFORMED ON THE CLIENT AUPRC DISTRIBUTION

Chi test	p-value
PCA data	0.04
No STD data - FD	0.99
No STD data - FedAvg	0.99
Glob STD data	0.04

($\mathrm{AUPRC}_{\mathrm{Ratio}} \simeq 1.6$). This highlights the critical importance of federated participation and data transformation in optimizing model outcomes.

A Chi-square test was conducted to validate these findings, confirming significant differences in the mean AUPRC distributions between standardized and transformed data (p-values between 0.1 and 0.15); see Table II. Additionally, data exchanged between nodes and the aggregator was analyzed. The results show that overhead traffic is directly influenced by the volume of data exchanged. Among the exchanged data types, model weights had the smallest size, averaging 224 bytes. In contrast, centralized models require the transmission of the entire device dataset, resulting in a significant overhead burden. However, the exact amount varies depending on device traffic and is not specified here. Other data transmitted during FL include feature sum and deviation vectors, each 56 bytes in size, which are essential for global standardization and principal component analysis (PCA). Basic traffic information is also communicated during the setup phase. Finally, the federated dropout (FD) algorithm requires additional data transfer, including the labeled logit vector (8 bytes) and model weights, further increasing the data exchange requirements.

The average data size for each client is around 140 MB. The average GPU utilization per example is 3.51 for centralized models and 2.15 for Federated approaches. The execution time for 1 MB is 5 seconds on average for the centralized model and 4.83 seconds on average for the Federated models. These values indicate that higher computational effort is required for validation on individual devices, but this enables a more distributed workload across the network, as previously demonstrated. Additionally, as depicted in Figure 6, when the example size increases, device GPU usage is lower for the Federated models than the centralized model, as previously mentioned.

VI. FUTURE CHALLENGES

The following section outlines future research goals and presents emerging challenges in network security. As network threats continue to evolve, it is imperative to develop methods for accurately classifying raw data, understanding the inner workings of these methods, and addressing critical security issues. A novel approach that combines computer vision techniques with Explainable AI (XAI) methods, such as saliency maps, offers promising advancements in this domain.

By converting raw data into visual representations, such as grayscale images constructed from hexadecimal arrays [22], and applying saliency maps to highlight key features [23], this method enhances the precision of malicious activity detection while clarifying the rationale behind machine learning decisions. Despite these advancements, research on integrating XAI and computer vision for cybersecurity analysis remains limited, presenting a significant research opportunity. Future work should investigate the effectiveness of different XAI methodologies combined with computer vision techniques to improve network anomaly detection and interpretability. This approach holds great potential for enhancing malware detection and security awareness as explainable AI develops.

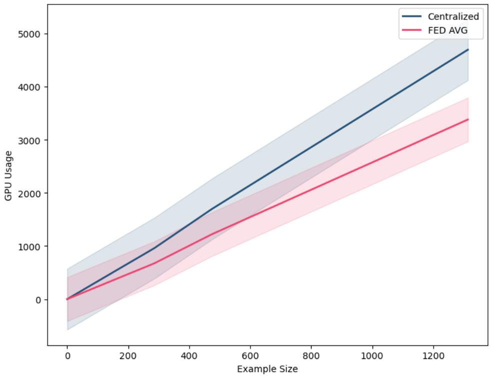


Fig. 6. The graph shows the GPU usage curve based on the test set size (number of examples). The blue curve represents the centralized model, while the pink curve represents Federated models. The pink curve demonstrates the lower GPU usage of the Federated models compared to the centralized model. The bars indicate the confidence interval at level 95%.

In addition to these advancements, our future research aims to expand the current framework to include multinomial cases, which we believe will significantly improve the model's diagnostic capabilities. This expansion will identify a broader range of malware categories, further strengthening cybersecurity defenses. We also plan to explore how federated learning can benefit from scenarios where only one or a small subset of clients encounters new malware. This research will focus on collaborative efforts within the federation to enhance security and learning outcomes, particularly through data transformation techniques that leverage computer vision approaches.

Finally, an important aspect not yet addressed in our framework is the *forgetting problem*, where models must continuously adapt to evolving data distributions and emerging patterns. We will investigate mechanisms to prevent, mitigate, or strategically use forgetting in machine learning models. This exploration aims to improve the model's resilience and versatility, enhancing its efficacy across various real-world applications.

VII. CONCLUSION

In conclusion, the IoT ecosystem, characterized by a vast network of interconnected devices that collect, exchange, and analyze data, is inherently vulnerable to malware attacks due to its simplicity and limited security features. To address this challenge, a federated approach for binary classification has been investigated, leveraging the decentralized nature of IoT devices and their computational capabilities to optimize learning while ensuring data security.

This study performed a comprehensive analysis using the IoT-23 dataset, which includes labeled instances of IoT malware infections. The results demonstrate that federated models, particularly those employing global standardization or principal component analysis (PCA), outperform traditional centralized approaches in the global area under the precision-recall curve (AUPRC) and exhibit lower variance.

The Federated Average approach, trained on globally standardized data, emerges as the most effective among the methods evaluated. It achieves a crucial balance between enhancing model performance and minimizing overhead traffic, making it a highly promising solution for federated learning applications in IoT environments. This approach improves security and supports efficient and scalable deployment across diverse IoT networks.

Acknowledgement

This work has been partially supported by the European Union under the Italian National Recovery and Resilience Plan (NRRP) of NextGenerationEU, a partnership on "Telecommunications of the Future" (PE0000001 - program "RESTART").

References

[1] K. K. Patel, S. M. Patel, and P. Scholar, "Internet of things-iot: definition, characteristics, architecture, enabling technologies, application & future challenges," *International journal of engineering science and computing*, vol. 6, no. 5, 2016.

[2] P. Asghari, A. M. Rahmani, and H. H. S. Javadi, "Service composition approaches in iot: A systematic review," *Journal of Network and Computer Applications*, vol. 120, pp. 61–77, 2018.

[3] M. K. Hasan, A. A. Habib, Z. Shukur, F. Ibrahim, S. Islam, and M. A. Razzaque, "Review on cyber-physical and cyber-security system in smart grid: Standards, protocols, constraints, and recommendations," *Journal of Network and Computer Applications*, vol. 209, p. 103540, 2023.

[4] A. A. Elngar, R. Chowdhury, M. Elhoseny, and V. E. Balas, *Applications of Computational Intelligence in Multi-Disciplinary Research*. Academic Press, 2022.

[5] F. Deldar and M. Abadi, "Deep learning for zero-day malware detection and classification: A survey," *ACM Computing Surveys*, 2023.

[6] N. Pachhala, S. Jothilakshmi, and B. P. Battula, "A comprehensive survey on identification of malware types and malware classification using machine learning techniques," in *2021 2nd International Conference on Smart Electronics and Communication (ICOSEC)*. IEEE, 2021, pp. 1207–1214.

[7] A. A. Almazroi and N. Ayub, "Deep learning hybridization for improved malware detection in smart internet of things," *Scientific Reports*, vol. 14, no. 1, p. 7838, 2024.

[8] A. Gaurav, B. B. Gupta, and P. K. Panigrahi, "A comprehensive survey on machine learning approaches for malware detection in iot-based enterprise information system," *Enterprise Information Systems*, vol. 17, no. 3, p. 2023764, 2023.

[9] Ö. A. Aslan and R. Samet, "A comprehensive review on malware detection approaches," *IEEE access*, vol. 8, pp. 6249–6271, 2020.

[10] A. Nascita, F. Cerasuolo, D. Di Monda, J. T. A. Garcia, A. Montieri, and A. Pescapè, "Machine and deep learning approaches for iot attack classification," in *IEEE INFOCOM 2022-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 2022, pp. 1–6.

[11] J. Jeon, J. H. Park, and Y.-S. Jeong, "Dynamic analysis for iot malware detection with convolution neural network model," *IEEE Access*, vol. 8, pp. 96 899–96 911, 2020.

[12] V. Rey, P. M. S. Sánchez, A. H. Celdrán, and G. Bovet, "Federated learning for malware detection in iot devices," *Computer Networks*, vol. 204, p. 108693, 2022.

[13] R. Taheri, M. Shojafar, M. Alazab, and R. Tafazolli, "Fed-iiot: A robust federated malware detection architecture in industrial iot," *IEEE transactions on industrial informatics*, vol. 17, no. 12, pp. 8442–8452, 2020.

[14] Q.-V. Pham, K. Dev, P. K. R. Maddikunta, T. R. Gadekallu, T. Huynh-The *et al.*, "Fusion of federated learning and industrial internet of things: A survey," *arXiv preprint arXiv:2101.00798*, 2021.

[15] M. Venkatasubramanian, A. H. Lashkari, and S. Hakak, "Iot malware analysis using federated learning: A comprehensive survey," *IEEE Access*, vol. 11, pp. 5004–5018, 2023.

[16] J. S.-P. Díaz and Á. L. García, "Study of the performance and scalability of federated learning for medical imaging with intermittent clients," *Neurocomputing*, vol. 518, pp. 142–154, 2023.

[17] E. Oja, H. Ogawa, and J. Wangviwattana, "Principal component analysis by homogeneous neural networks, part¡ cd02d35. gif¿: The weighted subspace criterion," *IEICE Transactions on Information and Systems*, vol. 75, no. 3, pp. 366–375, 1992.

[18] M. Venkatasubramanian, A. H. Lashkari, and S. Hakak, "Iot malware analysis using federated learning: A comprehensive survey," *IEEE Access*, 2023.

[19] H. Seo, J. Park, S. Oh, M. Bennis, and S.-L. Kim, "16 federated knowledge distillation," *Machine Learning and Wireless Communications*, p. 457, 2022.

[20] W. Issa, N. Moustafa, B. Turnbull, N. Sohrabi, and Z. Tari, "Blockchain-based federated learning for securing internet of things: A comprehensive survey," *ACM Computing Surveys*, vol. 55, no. 9, pp. 1–43, 2023.

[21] H. R. Sofaer, J. A. Hoeting, and C. S. Jarnevich, "The area under the precision-recall curve as a performance metric for rare binary events," *Methods in Ecology and Evolution*, vol. 10, no. 4, pp. 565–577, 2019.

[22] X. Ma, Z. Dai, Z. He, J. Ma, Y. Wang, and Y. Wang, "Learning traffic as images: A deep convolutional neural network for large-scale transportation network speed prediction," *Sensors*, vol. 17, no. 4, p. 818, 2017.

[23] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps," *arXiv preprint arXiv:1312.6034*, 2013.

Appendix

The IoT-23 dataset, collected by Avast Software, Prague, consists of twenty-three captures of IoT network traffic from 2018 to 2019. Each scenario involves executing specific malware samples on a Raspberry Pi using various protocols. The dataset includes original .pcap files, corresponding Zeek *conn.log* files, and*conn.log.labelled* files with additional labeling columns. Due to large traffic volumes, .pcap files are rotated every 24 hours, sometimes resulting in varying capture durations.

These labels are one for the binary classes (Malicious, Benign), the other for the category of malware. In this work, only the binary labels are considered, and only the following features are considered:

- *duration*: How long the connection lasted
- *origin bytes*: The number of payload bytes sent by the originator. This is taken from sequence numbers for TCP and may be inaccurate (e.g., due to large connections).
- *missed bytes*: Indicates the number of bytes missed in content gaps, representing packet loss. A value other than zero will normally cause protocol analysis to fail but some analysis may have been completed before the packet loss.
- *originator packets*: Number of packets that the originator sent.
- *originator IP bytes*: Number of IP level bytes sent by the originator
- *responder packets*: Number of packets sent by the responder
- *responder IP bytes*: Number of IP level bytes sent by the responder (as seen on the wire, taken from the IP *total length* header field).

Of course, because of the nature of the experiment, dataset sizes vary significantly, with smaller sets comprising approximately four examples while larger sets contain over 100,000 samples. For this reason, the datasets are considered if they have at least 100 examples.

A Convolutional Neural Network to Locate Unbalance in Turbomachinery Supported by AMBs

Giovanni Donati
Department of Information Engineering
University of Florence
Florence, Italy
giovanni.donati@unifi.it

Michele Basso
Department of Information Engineering
University of Florence
Florence, Italy
michele.basso@unifi.it

Marco Mugnaini
Department of Information Engineering and Mathematics
University of Siena
Siena, Italy
marco.mugnaini@unisi.it

Chiara Camerota
Department of Information Engineering
University of Florence
Florence, Italy
chiara.camerota@unifi.it

Abstract— **Due to the complex structure of turbomachinery systems, the process of fault detection assumes paramount importance, in particular rotor unbalance faults are particularly risky and common. This research paper introduces an innovative and straightforward approach to locate rotor unbalance faults for turbomachinery supported by Active magnetic bearings (AMB) exploiting the AMB sensors and utilizing Deep Learning techniques (1D Convolutional Neural Networks). The main goal of this study is to develop a fault dictionary, built using fault signatures derived from position sensor signals, and a classifier specialized in locating the unbalance faults in turbomachinery supported by AMBs, that generally occur in the turbomachine impellers. These are the most prevalent unbalance faults that affect turbomachinery systems and that commonly impact the performance of AMB systems. The effectiveness of this approach is demonstrated through a case study involving an expander-compressor supported by two active magnetic bearings in the oil and gas field. Five distinct fault classes are considered, and the neural network fault classifier achieves an impressive accuracy rate of 98% on the test dataset.**

Keywords—Active Magnetic Bearings, AMB, Rotor fault, Unbalance Fault, Convolutional Neural Networks

I. INTRODUCTION

Active Magnetic Bearings (AMBs) are becoming increasingly important in a wide range of rotating machinery applications. These applications can vary from small turbo molecular pumps used in medicine to large megawatt-scale compressors used in the oil and gas industry. Unlike traditional contact bearings, AMBs offer significant advantages. One key benefit is their ability to eliminate friction and wear by levitating the rotor with respect to the stator components. This enables higher rotational speeds, improves system efficiency, and eliminates the need for complex lubrication systems typically found in traditional bearings. However, due to their inherently unstable nature, AMBs require a feedback control system to operate. This closed loop comprises different components, including controller, position sensors, amplifiers and actuators. A detailed description of the structure of AMB systems can be find in [1]. The AMB system component choice determines the nominal stiffness and damping characteristics of the bearings. These properties establish the overall stability and performance of the closed-loop system. Beyond the already mentioned advantages, AMBs can be exploited for fault location and even for fault compensation. In fact, the sensors on board AMB systems allow for continuously monitoring the behavior of the rotor during operations and provide information useful for detecting faults, ensuring safe and reliable operation.

The early detection of failures in AMB systems offers the opportunity to employ safety strategies that leverage the closed-loop architecture to adapt the system and respond to the detected faults in real-time. The active nature of AMB controllers allows for dynamic adjustments to the bearing behavior, enhancing the system chances of survival exploiting an internal information processing. Consequently, fault tolerant AMB systems have been developed to effectively manage malfunctions, utilizing for example reconfigurable control strategies [2]. As a result, numerous research studies have focused on developing methods to identify failures associated with the rotor or the electrical and electronic components.

A common practice to identify faults involves the use of system observers that compare the system state with the expected one related to the normal operational condition to identify any anomalies. A different approach is the simulation-before-test technique. This method involves creating a fault dictionary through simulations of a specific system [3], which accumulates instances of fault signatures, and these signatures serve as training data for a classifier capable of recognizing various faults, as elucidated in reference [4]. Traditionally, fault diagnosis has revolved around analyzing signals in the time and frequency domains. A similar methodology is put forth by Jing *et al.* in [5], who proposed a feature-based learning and fault diagnosis method for gearbox condition monitoring.

However, some advanced techniques exploit a fault dictionary composed of images generated from time-domain signals. These representations serve as the base for training convolutional neural networks (CNNs) specifically designed for the purpose of detecting and recognizing faulty conditions. This method applied to AMB systems was pioneered by Yan *et al.*, who employed sensor signals to construct orbits for detecting mechanical failures, as outlined in reference [6].

This research focuses on detecting and locating unbalance faults in the impellers of a turbomachine supported by AMBs in operating condition, that is one of the most common situations that occurs in a turbomachine [7] using AMBs as an identification and diagnosis tool for the system. In the context of this work, only single unbalance faults have been considered for simplicity, but the fault classes can be extended. The proposed work introduces a novel approach which aims at locating impeller unbalance faults through the application of CNN by dividing the impeller into different circular sectors and assigning to each region a different class label. The approach is demonstrated in a specific case in this paper but is a general procedure which allows for enhancing the accuracy of unbalance location by increasing the number of circular sectors. This work should be considered as an extension of the authors' work [4] which had the objective of identifying electrical faults in AMB systems. The new approach can be used together with the one previously developed.

To construct the fault dictionary, the position sensor electrical signals are harnessed as sources to generate generalized orbits, which are then transformed into discrete black and white images. The properly trained CNN, known for its proficiency

in image classification has the dual objective of identifying the defective impeller and precisely determining the location of the unbalance fault within said impeller. More specifically, the approach followed employed 1D CNNs, which exhibit great performance, as evidenced in [8], working with grid-like topological data. Furthermore, by employing computationally efficient image processing techniques, that is Adam optimization [9], in conjunction with efficient neural networks, it becomes feasible to construct an automatic online diagnostic system with minimal computational overhead.
The paper structure is the following: Section II describes the fault dictionary building process. Section III describes the proposed model of classifier trained with the fault dictionary described in the previous section. Section IV presents a case study and the obtained results. Conclusion follows.

II. Fault Dictionary Building

A. System Modeling

An AMB consists of a pair of opposing electromagnets, which exert attractive forces on a ferromagnetic object, in this case, the rotor. Their objective is to maintain the rotor in the center of the air gap. Typically, a turbomachine is equipped with no fewer than five AMBs, consisting of four dedicated to managing the radial dynamics of the rotor (two for each radial bearing) and one dedicated to controlling the axial dynamics. This study focuses exclusively on the radial dynamics of the system since radial rotor dynamics tend to be inherently more complex than its axial counterpart.

Since AMBs are inherently unstable components, they are always inserted into a stabilizing closed-loop system. Fig. 1 presents a block diagram illustrating the closed-loop configuration of an AMB system. This system includes amplifiers that drive in current the magnetic bearings, position sensors accompanied by their associated conditioning electronics, as well as a controller.

The role of the position sensors is to continuously monitor the rotor position, providing real-time data. The controller leverages these position signals to compute the required control inputs for the actuators, which drive the magnetic bearings. The main objective of this closed-loop system is to consistently maintain the rotor in a suspended state at the center of the air gap. This is achieved by determining the optimal control currents to drive each individual AMB.

To model the closed-loop AMB system, a state-space formulation was developed for each component. The rotor radial state-space model was specifically constructed using a finite element method (FEM) using Timoshenko modeling [10]. Additionally, linear state-space models were used to describe the electrical components of the AMB system, considering their unique characteristics and specifications. Combining the rotor state space and the electrical component models, a closed-loop system state space was found:

$$\frac{d}{dt}\begin{bmatrix} X_R \\ X_B \end{bmatrix} = \begin{bmatrix} A_R & B_R C_B \\ B_B C_R & A_B \end{bmatrix}\begin{bmatrix} X_R \\ X_B \end{bmatrix} + \begin{bmatrix} B_R \\ [0] \end{bmatrix} F_{ext} =$$

$$= A\begin{bmatrix} X_R \\ X_B \end{bmatrix} + BF_{ext} \tag{1}$$

where X_R and (A_R, B_R, C_R) represent the state and the matrices of the rotor model, X_B and (A_B, B_B, C_B) are the state and matrices of the second state space of the electrical models and F_{ext} the external forces acting on the system. Equation (1) allows for studying the performance and dynamic behavior of the AMB system model.

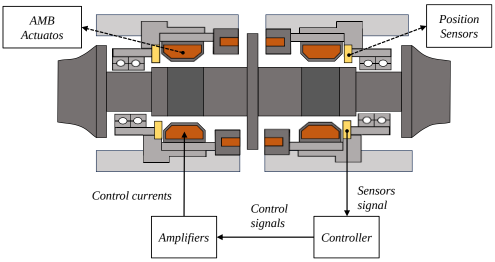


Fig. 1. Schematic of an AMB system structure.

In high-speed turbomachine, the main contribution of the term F_{ext} is the unbalance force. Unbalance forces are synchronous with the rotor speed and arise from the existence of unbalanced masses relative to the rotor axis of rotation. These unbalanced masses result from inherent manufacturing imperfections or arise from wear. When considering a fixed rotor speed Ω along a given axis x, an unbalance force is added to F_{ext}, which takes the following expression:

$$F_{un_x} = \Omega^2 U \cos(\Omega t + \phi) \tag{2}$$

where ϕ is the phase of the unbalance with respect to other axis and U is the unbalance magnitude in kgm. To exactly reproduce the dynamic behavior of a real system, each machine is experimental identified to accurately assess the unbalance and to fine tune the state-space model (1), as for example accurately described in [11].

B. Unbalance Fault

Due to the complex mechatronics nature of AMB systems, failures can manifest in different ways, software, electrical, or mechanical, which can impact system performance differently. Lijesh *et al.* in [12] summarizes common malfunction causes, their occurrence, and severity. These faults can lead to high-speed rotor contact with the housing, exposing plant safety. To mitigate these risks, measures like redundancies, quality control, individual actions, and various control strategies are essential. Active fault diagnostics and corrections are also valuable. This work focuses on unbalance faults, whose action on the AMB closed-loop system can be modeled by (2). Even if the rotor is balanced in the manufacturing process, a residual unbalance always remains. Following an identification process this unbalance is measured and characterizes the particular turbomachine. During normal operation, thanks to AMB levitation, wear mainly occurs only on the impeller blades. Solid particle erosion is one of the main reasons causing the blades wear failure. Wang *et al.* in [13] analyzed the primary influencing factors on erosion such as the particle characteristics, environmental condition, and materials characteristics. Normally, the worn blades must be localized and repaired in an impeller. Every worn blade act as an unbalance fault that contributes to the vibrations of all the AMB closed-loop system.

This work aim is to localize the faulty impeller and the position of the faulty blade, considering a single fault at a time. More specifically, the localization is intended as the identification of the phase ϕ between the native unbalance ($\phi = 0^o$) of the particular turbomachine and the faulty blade. To this aim, the impeller is divided into equal circular sectors and the aim of this work is to locate the sector that contains the faulty blade. Fig. 2 summarizes this idea. In this work, for demonstration purposes, only two circular sectors (β_1 and β_2) per impeller were considered but the sector number can be arbitrarily increased if needed.

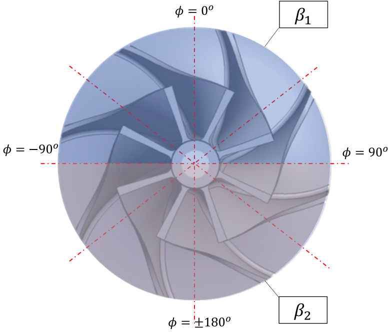


Fig. 2. Impeller sector division.

C. Fault Dictionary

Steady-state signals are exploited to construct images that serve as fault signatures which act as input data to train a classifier. In this work position signals from sensors are considered, that are always available in an AMB system.
Considering that all signals inherently present periodic behavior when the turbomachine operates at a fixed rotational speed, under steady conditions, it becomes feasible to organize these time signals into couples, by expressing the signal associated with one of the two orthogonal control axes of a radial AMB as a function of the other. In this process, time is no longer treated as an independent variable, and one generalized signal trajectory or orbit is generated for each radial bearing. In the presence of a fault these orbits change assuming specific and peculiar shapes corresponding to the related fault. Therefore, the collection of the different orbit signatures in the presence of different faults can be used as a fault dictionary.

The fault orbits were derived through the utilization of a simulation tool capable of automatically building the entire fault dictionary modeling the fault conditions, as described in [3]. This simulation tool has been implemented within the Matlab-Simulink environment and enables the accomplishment of Monte Carlo simulations by using specific probability distributions to simulate the parameter variation ranges of the AMB system components or by adding noisy signals. The developed simulation tool was validated by comparing the results obtained with commercial software (MADYN 2000). To create reference orbits linked to the non-faulty condition, machine operations at a constant rotor speed were simulated while varying the component parameters within their specified tolerance ranges and accounting for normal level of noise. The dictionary is completed by collecting the orbit orbits linked to faulty conditions obtained simulating the different impellers unbalance faults by randomly varying different unbalance phases and magnitudes. Variations within component parameter tolerance ranges define confidence regions containing the reference orbits representing the fault-free condition. Any deviations of the orbits outside these regions signify a fault occurrence. The orbit signatures were represented by images with a fixed resolution. Black and white images were generated through the Matplotlib library in Python. Utilizing images directly from the available signals to train a classifier enhances the system capability to comprehend the AMB system state. This approach provides a broader perspective and improves the precision of captured information.

III. CONVOLUTIONAL NEURAL NETWORKS

The unbalance fault localization issue can be considered as a classification problem. Addressing the sensor orbit images, a CNN-based classifier was developed. This model is tailored to handle grid-like topological data, encompassing a wide range of data types that can be conceptualized as either 1D or 2D grids, including time-series data and images. AlexNet, introduced by Krizhevsky *et al.* in [14], emerged as the pioneer of modern deep CNNs in the realm of image classification. The CNN architecture comprises sequential layers, organized as follows: convolutional layers, triggered by a chosen activation function, pooling layers, and fully connected layers. A convolutional layer involves sliding a filter/kernel across the input data, performing convolutions. Depending on patterns and spatial information, a filter can extract various features, yielding a feature map as its output. The activation function serves to introduce non-linearity into the network, with ReLU and its derivatives being the most prevalent choices [15]. The pooling layer plays a dual role in neural networks; it extracts essential information from the feature map while concomitantly diminishing computational complexity. For instance, a widely employed technique is Max pooling [16], wherein the highest value within a given region signifies the entire content of that region. Finally, the feature map is flattened into a one-dimensional array and fed into the fully connected layers for classification and prediction. The choice of the final function depends on the particular task. For multivariate classification problems, one common choice is the SoftMax function [17]. To evaluate the error, i.e., the disparity between the prediction and the actual value, a loss function is employed. Once again, the choice of this function is contingent on the nature of the problem being addressed. An optimization algorithm is then utilized to minimize the loss function and refine the accuracy of the network predictions. A frequently used algorithm is the Adam Optimizer and its variants [9]. In summary, the input data undergoes a series of transformations through the convolutional layer, followed by summarization, flattening, and prediction computation. This iterative process involves updating the network weights to minimize the loss function, enhancing the accuracy of predictions with each iteration.

IV. CASE STUDY AND RESULTS

A. Case Study Description

As a case study to assess the performance of the proposed diagnosis method a medium size expander-compressor supported by AMBs for oil and gas application was considered. The rotor has a mass of about 220 kg and a length of about 1.233 m. For what concerns the electrical part, the controller structure was composed by augmented PIDs. Additionally, other components included switching pulse width modulation (PWM) amplifiers coupled with heteropolar AMB actuators and inductive position sensors operating within a 2.5 kHz bandwidth. Under nominal conditions, the system features a nominal rotation speed of 6920 rpm and a native unbalance of 1.45×10^{-4} kgm positioned at the center of mass of the rotor.

For this case study, tolerance parameters encompassed a 2% variation in sensor sensitivities, sensor biases of 1 μm, and Gaussian white noise characterized by a standard deviation of 1 μm. Regarding the actuators, a 2% tolerance of the DC gain was taken into consideration. Since the number of impellers of the system is two, the fault classes considered were four, considering two circular sector per impeller (β_1 and β_2), as shown in Fig. 2. The five classes and the number of examples per class are summarized in Table I. The faulty conditions that were taken into account to form the classes of the fault dictionary were simulated with a Monte Carlo analysis by varying the fault unbalance magnitude in the range $[0, 6.00 \times 10^{-4}]$ kgm and its phase in the prescribed range using a uniform distribution.

TABLE I.

Class Name	***Example Class Size***
Nominal Condition	3000
Expander fault in β_1	2000
Expander fault in β_2	2000
Compressor fault in β_1	2000
Compressor fault in β_2	2000

B. Proposed CNN Model

As anticipated, the CNN classifier is designed with the specific aim of identifying five classes: the Nominal Condition, two unbalance faults on the expander impeller and two unbalance faults on the compressor impeller. The classifier structure is illustrated in Fig. 3. It comprises a series of stacked convolutional, pooling, and fully connected layers. The input to the network consists of a grayscale image depicting two orbits from sensors, one for each AMB, while the output gives the probability distribution across the five classes. As previously mentioned, when the activation function triggers a convolutional layer, it extracts key features utilizing filters. These outputs are then aggregated through max-pooling and subsequently classified by the fully connected layers. All convolutional layers employed are 1D, simplifying network complexity and computational load. They have a kernel dimension of 3×3, and a stride dimension of 1×1, activated by a ReLu function [15]. The max-pooling operation is executed with a stride dimension of 2×2. The initial convolutional layer features one input channel and produces eight output channels. The second convolutional layer mirrors this configuration with eight input and output channels. This approach is employed to expand and then consolidate information, emphasizing critical features. This process also serves to mitigate gradient explosion following a max-pooling layer. Subsequently, a batch normalization step is implemented to preclude dropout usage, as demonstrated in [18]. This ensures that initialization does not unduly impact results. The final two convolutional and max-pooling layers are helpful in amalgamating and distilling output from preceding layers, extracting only the most significant features, including intricate patterns and structures. Subsequently, the fully connected component commences. Initially, the data vector undergoes a transformation into a one-dimensional tensor, after which three linear layers are trained. These layers progressively reduce the vector dimension until arriving at the five classification classes. In this study, an Adam Optimizer [9] was employed to optimize algorithm efficiency, balancing computational resources and memory usage. The architectural implementation was executed utilizing the PyTorch libraries.

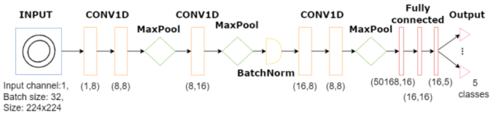


Fig. 3. CNN proposed model structure, the numbers below the layer represent the input and output channels.

C. Results

The dataset, obtained as described in the previous section, is composed of 3000 examples of the fault free condition and 2000 examples for each of the other 4 fault classes. The training set includes 80% of the examples and the test and a validation set collects 10 % of the remaining examples each. Each image had a size of 224×224 pixel.

The model was trained using the pre-processed training set (employing a batch size of 32) for a total of 300 epochs, using a learning rate of 10^{-6}. This training process took approximately 4 hours, utilizing a GPU NVIDIA RTX A6000 with 48 GB GDDR6. Validation of the proposed Convolutional Neural Network (CNN) was carried out using the test dataset, where it achieved an accuracy of approximately 98%. Simultaneously, the training dataset yielded an accuracy of around 99%. The obtained interference time is 2×10^{-4} s, thereby validating the feasibility of implementing the proposed model in real-time applications. These results are graphically represented in Fig. 4 and Fig. 5 showcasing respectively test set accuracy and training set performance.

Fig. 6 illustrates the test and accuracy ratio, which shows the relationship between the training and test accuracies depicted in Fig. 4 and Fig. 5. The declining trend observed in this graph suggests that the estimated parameters of the CNN remain unaffected by overfitting. This observation implies that the model has achieved a good level of generalization, indicating its capability to effectively apply learned patterns and make accurate predictions when confronted with previously unseen data.

Furthermore, Fig. 7 presents the resulting confusion matrix, in which the horizontal axis represents the actual faults, and the vertical axis represents the predicted faults. The diagonal elements represent the percentage of accurate predictions, while values above the diagonal denote false positives, and those below signify false negatives. As depicted in Fig. 7, the confusion matrix displays a notably diagonal pattern, with even the lowest values hovering around 98%. Most misclassifications primarily are associated with minor parametric deviations from the fault-free condition. The achieved accuracy in recognizing these faults, even with small parametric variations, is quite satisfactory for two key reasons. First, the regions of parameter variation related to faults closely border the tolerance regions. Second, the limited image resolution contributes to some classes being recognized less accurately, specifically those associated with orbits that exhibit minor differences compared to the reference orbits when subjected to slight parameter variations beyond the tolerance, as there are no clear-cut boundaries between faulty and fault-free conditions. However, it's worth noting that false positives or negatives in these borderline situations do not have significant consequences.

Similar accuracy was also achieved in [4] where a 3D convolutional neural network was implemented to identify electrical faults, and in [19] where several convolutional neural networks were tested to identify simpler mechanical faults.

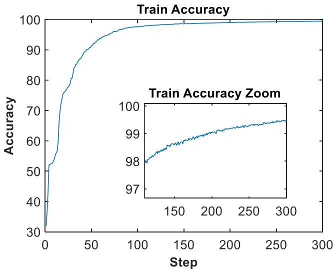


Fig. 4. Train accuracy for each epoch.

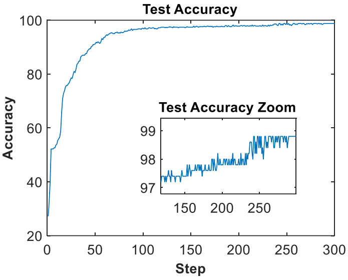


Fig. 5. Test accuracy for each epoch.

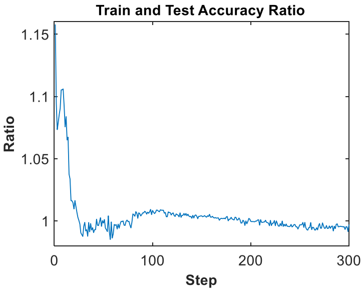
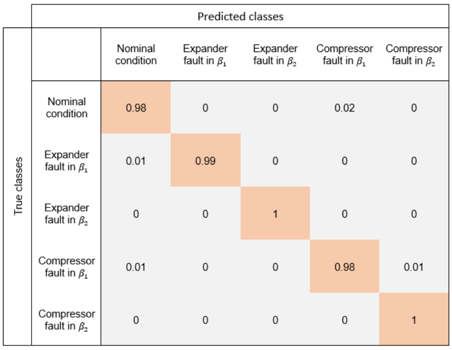


Fig. 6. Train and test accuracy ratio plot.

V. CONCLUSION

Active Magnetic Bearings are gaining prominence across various rotating machinery applications owing to their exceptional performance and the elimination of cumbersome lubrication systems. However, their operation necessitates a complex closed-loop mechatronic system. To ensure safe and reliable operation, the deployment of fault detection and diagnostic systems becomes imperative.

The novel approach presented here for fault diagnosis leverages sensor signals within an AMB system to build a fault dictionary with the aim of training a classifier – a straightforward convolutional neural network. This classifier has been specifically trained to locate the unbalance faults in the turbomachine impellers. The proposed classifier demonstrates remarkable accuracy and generalizability, all while requiring a modest amount of data. Importantly, this approach can be easily readily extended to address other fault types and can be applied to a variety of AMB-supported systems.



True classes \ Predicted classes	Nominal condition	Expander fault in β_1	Expander fault in β_2	Compressor fault in β_1	Compressor fault in β_2
Nominal condition	0.98	0	0	0.02	0
Expander fault in β_1	0.01	0.99	0	0	0
Expander fault in β_2	0	0	1	0	0
Compressor fault in β_1	0.01	0	0	0.98	0.01
Compressor fault in β_2	0	0	0	0	1

Fig. 7. Confusion matrix of test set.

REFERENCES

[1] Srinivas R. S., Tiwari R., Kannababu C, “Application of active magnetic bearings in flexible rotordynamic systems–A state-of-the-art review,” Mechanical Systems and Signal Processing, vol. 106, pp. 537-572, 2018.

[2] Tsai N. C., King Y. H., Lee R. M, “Fault diagnosis for magnetic bearing systems. Mechanical systems and signal Processing,” vol. 23(4), pp. 1339-1351, 2009.

[3] M. Basso, G. Donati and M. Mugnaini, "Smart Fault Dictionary for Active Magnetic Bearings Systems," 2023 IEEE International Workshop on Metrology for Industry 4.0 & IoT (MetroInd4.0&IoT), Brescia, Italy, 2023, pp. 360-365, doi: 10.1109/MetroInd4.0IoT57462.2023.10180183.

[4] Donati G., Basso M., Manduzio G. A., Mugnaini M., Pecorella T., Camerota C., “A Convolutional Neural Network for Electrical Fault Recognition in Active Magnetic Bearing Systems,” Sensors, vol- 23, no. 16, 7023, 2023.

[5] Jing L., Zhao M., Li P., Xu X. A convolutional neural network-based feature learning and fault diagnosis method for the condition monitoring of gearbox. Measurement, , vol. 111, pp. 1–10, 2017

[6] Yan X., Zhang C. A., Liu Y., “Multi-branch convolutional neural network with generalized shaft orbit for fault diagnosis of active magnetic bearing-rotor system,” Measurement, vol. 171, 108778, 2021.

[7] Zabihihesari A., A. Shirazi, F., Riasi A., Mahjoob M., Asnaashari E., “Simulation-based Vibration Sensor Placement for Centrifugal Pump Impeller Fault Detection,” Journal of Computational Applied Mechanics, vol. 51, no. 1, pp. 72-80, 2020.

[8] Junior R. F. R., Dos Santos Areias I. A., Campos M. M., Teixeira C. E. da Silva L. E. B., Gomes G. F., “Fault detection and diagnosis in

electric motors using 1d convolutional neural networks with multi-channel vibration signals," Measurement, vol. 190, 110759, 2022.

[9] Kingma D. P., Ba J., "Adam: A Method for Stochastic Optimization*," ArXiv. /abs/1412.698*, 2014.

[10] Friswell M. I., Penny J. E., Garvey S. D., Lees A. W., Dynamics of rotating machines. Cambridge university press, 2010.

[11] Wroblewski A. C., Sawicki J. T., Pesch, A. H., "Rotor model updating and validation for an active magnetic bearing based high-speed machining spindle," Journal of Engineering for Gas Turbines and Power, vol. 134, no. 12, 122509, 2022.

[12] Lijesh K. P., Hirani H, "Failure Mode and Effect Analysis of Active Magnetic Bearings," Tribology in Industry, vol. 38.1, 2016.

[13] Wang G., Jia X., Li J., Li F., Liu Z., Gong B., "Current state and development of the research on solid particle erosion and repair of turbomachine blades," In Re-engineering Manufacturing for Sustainability: Proceedings of the 20th CIRP International Conference on Life Cycle Engineering, Singapore 17-19 April, pp. 633-638, 2013.

[14] Krizhevsky Alex, Ilya Sutskever, Geoffrey E. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM,* vol. 60, no. 6, 2017, pp. 84-90.

[15] Agarap A. F., "Deep learning using rectified linear units (relu)," arXiv preprint arXiv:1803.08375, 2018.

[16] J. Nagi et al., "Max-pooling convolutional neural networks for vision-based hand gesture recognition," 2011 IEEE International Conference on Signal and Image Processing Applications (ICSIPA), Kuala Lumpur, Malaysia, 2011, pp. 342-347, doi: 10.1109/ICSIPA.2011.6144164.

[17] Liu W., Wen Y., Yu Z., Yang M., "Large margin softmax loss for convolutional neural networks," arXiv preprint arXiv, 1612.02295, 2016.

[18] Ioffe S., Szegedy C., "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," *ArXiv. /abs/1502.0*.

[19] Y. Hu, O. W. Taha, K. Yang, "Fault Detection in Active Magnetic Bearings Using Digital Twin Technology" Applied Sciences, Vol. 14, no. 4: 1384, 2024. doi: 10.3390/app1404138

sensors

MDPI

Article

A Convolutional Neural Network for Electrical Fault Recognition in Active Magnetic Bearings Systems

Giovanni Donati[1], Michele Basso[1], Graziano A. Manduzio[3], Marco Mugnaini[2], Tommaso Pecorella[1] and Chiara Camerota[1]

[1] Department of Information Engineering, University of Florence, Via di Santa Marta 3 50139 Florence, Italy; michele.basso@unifi.it (M.B.); giovanni.donati@unifi.it (G.D.); chiara.camerota@unifi.it (C.C.).

[2] Department of Information Engineering and Mathematics, University of Siena, Siena, Italy; marco.mugnaini@unisi.it.

[3] Department of Information Engineering, University of Pisa, Pisa, Italy; graziano.manduzio@unipi.it.

Abstract: Active magnetic bearings are complex mechatronic systems that consist of mechanical, electrical, and software parts, unlike classical rolling bearings. Given the complexity of this type of system, fault detection is a critical process. This paper presents a new and easy way to detect faults based on the use of a fault dictionary and Machine Learning. In particular, the dictionary was built starting from fault signatures consisting of images obtained from the signals available in the system. Subsequently, a convolutional neural network was trained to recognize such fault signature images. This work concentrates on recognizing the most frequent electrical faults that typically affect position sensors and actuators. This new method permits, in a computationally convenient way, that can be implemented in real time, to determine which component has failed and what kind of failure has occurred. Therefore, this fault identification system allows for determining, in the event of a fault, which countermeasure to adopt in order to enhance the reliability of the system. The performance of the proposed method is assessed by means of a case study concerning a real turbomachine supported by two active magnetic bearings for the oil and gas field. Seventeen fault classes have been considered and the neural network fault classifier reached an accuracy of 93% on the test dataset.

Keywords: Active Magnetic Bearing, AMB, Fault analysis, Convolutional Neural Networks, Fault dictionary, Rotordynamics.

1. Introduction

Citation: To be added by editorial staff during production.

Academic Editor: Firstname Lastname

Received: date
Revised: date
Accepted: date
Published: date

Active Magnetic Bearings (AMBs) are being increasingly used across a broad range of rotodynamic applications, ranging from small turbo molecular pumps for medical applications to large compressors in the megawatt range for the oil and gas field. Compared to classic contact bearings, AMBs offer several significant advantages. AMBs achieve a significant reduction in friction and in the associated wear by levitating the rotor relative to the stator parts. The elimination of friction also enables higher rotation speeds, greater system efficiency, and eliminates the need for cumbersome lubrication systems that are typically required for traditional bearings. Due to the inherently unstable nature of AMBs, a stabilizing feedback control is needed for proper functioning. Therefore, an AMB is a complex mechatronic system comprising various components, including the controller, position sensors and actuators. The synthesis of the controller is a critical aspect of AMB systems. Various controller structures and design processes can be used, as reviewed for example in [1] or in [2]. Among these, augmented PIDs are probably the most widespread choice in industry because of their versatility, accuracy, efficiency, and cost-effectiveness. All the controllers of AMB systems rely on precise measurements of the rotor position, for which a common choice is to use inductive or eddy current sensors, even if recently self-sensing sensors and optical sensors have also been introduced [3]. The effect of the choice of sensors is discussed for example here [4]. About the other components, PWM is commonly used for driving the electromagnets that make the rotor levitate. The behavior of all the components in the loop contributes to determining the stiffness and damping of the bearings, that are directly related to the stability and performance of the closed-loop system. To improve the robustness of such a complex system, a fault detection and diagnostic system is advisable to ensure safe and reliable operation. As a result, several studies have developed methods to detect failures associated with the rotor or with the electrical and electronic components, as for example in [6] and in [7]. Failure detection in AMB system enables also to adopt safety strategies exploiting their closed loop architecture that can adapt to cope with the detected fault. In fact, due to their active nature, controllers can dynamically adjust the bearings behavior in real-time. Exploiting the power of internal information processing, an AMB system can enhance its survival chances and reliability. For this reason, fault tolerant AMB systems have been developed that can manage malfunctions, for example, using a reconfigurable control, as described in [6] or in [7].

For fault diagnosis systems, usually, the actual system behavior is compared to the expected one in nominal condition to identify a faulty system condition. Observers of the system, as described in [5], are typically used.

Another common approach for identifying faults is the simulation-before-test technique. It involves constructing a fault dictionary from simulations of a particular plant, which collects examples of fault signatures that can be used to train a classifier capable of recognizing different faults, as described for example in [8].

Traditionally, fault diagnosis has been based on analyzing signals in the time and frequency domains. However, this paper proposes an approach that exploits a fault dictionary made by images of signals in time-domain to train a simple convolutional neural network that has the goal to recognize the AMB system faulty conditions. Taking the available electrical signals as sources, generalized orbits are built and converted into discrete 2D images that are used to fill the fault dictionary, with a technique that generalizes and extends the approach proposed by Xunshi, who used the sensors signals to build orbits to detect mechanical failures [9]. A similar method is proposed by Jing et al., who in their paper proposed a feature-based learning and fault diagnosis method for gearbox condition monitoring [10]. The fault features were obtained by using a simulation tool, developed by some of the authors [11], capable of automatically building the entire fault dictionary, once the fault conditions are modeled. In the context of this work, only single electrical parametric faults have been considered for simplicity, but the fault classes can be extended including e.g. also mechanical faults as in [12].

To exploit the knowledge stored in the fault dictionary, this paper proposes a classifier based on a convolutional neural network (CNN), well-suited for image classification, [13, 14] trained with the fault signature examples. In the case of system fault, the trained convolutional neural network has the aim of identifying the faulty component and the type of fault that has occurred.

Compared to other methods proposed in the literature that rely on analyzing signals in the time domain, the proposed approach, losing the time dependence, offers a novel solution to detect and locate faults, making full use of the potential of smart systems like AMBs. Moreover, with the help of computationally efficient image processing and simple neural networks, this automatic online diagnostic system can be developed without excessive computational cost. Such systems could be exploited in advanced prognostic maintenance systems to enhance balance of plant capabilities over time as described in [15-17].

The paper structure is the following: Section 2 describes the AMB modeling adopted to build a simulation tool. Section 3 presents the developed fault dictionary by which a classifier can be trained. Section 4 describes the proposed model of image classifier. Section 5 presents a case study. Section 6 shows the case study results; conclusion follows.

2. System Modeling

3.1. AMB Background

An AMB, applied to a turbomachine, has the purpose of making the rotor levitate with respect to the stator. Normally, a turbomachine is equipped with at least five AMB control axes: four for the rotor radial dynamics (two per each radial bearing) and one for the axial one. Only the radial dynamics of the system has been considered since radial rotor dynamics are normally more complex than axial ones and for this reason has been chosen as object of this work. An AMB is composed of two opposed electromagnets that attract a ferromagnetic object, in this case the rotor, and try to maintain it in the center of the air gap. A radial bearing is composed of two AMB control axes that are able to attract the rotor in any direction in a plane. Figure 1 describes a classical cross section of a radial AMB bearing in a heteropolar configuration, as described in [2].

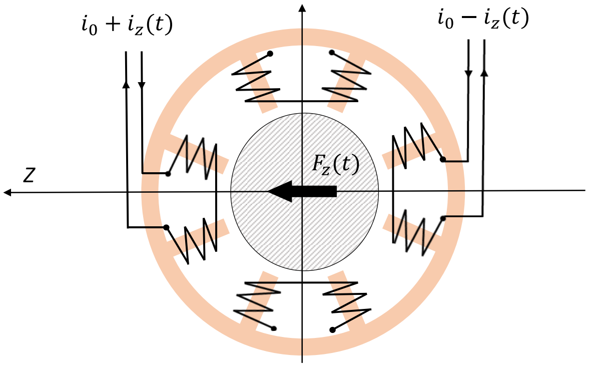


Figure 1. A classical cross section of a radial AMB in a heteropolar configuration.

The two opposed electromagnets are driven with a differential control current. Considering them, along an axis z, a linear relation can be found [18]:

$$F_z = k_i i_z - k_s Z \quad (1)$$

where F_z is the magnetic force exerted on the rotor along the axis z, Z is the displacement along the same axis with respect to the nominal working conditions and the parameters k_i and k_s respectively called electrical gain and negative stiffness depend on the geometrical parameters of the bearings and on the nominal operating conditions, that are the nominal air gap s_0 and the bias current i_0. Equation (1) is the more accurate the more the working conditions are close to the nominal ones. Because of the negative stiffness k_s, AMBs are inherently unstable. Hence, they are always inserted in a stabilizing closed loop system. Figure 2 shows a block diagram of an AMB system closed loop. This system comprises amplifiers for driving the magnetic bearings, position sensors comprising the related conditioning electronics, and a controller. The position sensors constantly monitor the rotor position, while the controller utilizes these signals to calculate the necessary control signals for the actuators, which drive the magnetic bearings. The goal of the closed loop system is to maintain the rotor in a levitated state at the center of the air gap by determining the appropriate currents $\bar{\imath}$ for each AMB, where $\bar{\imath}$ is the control currents vector.

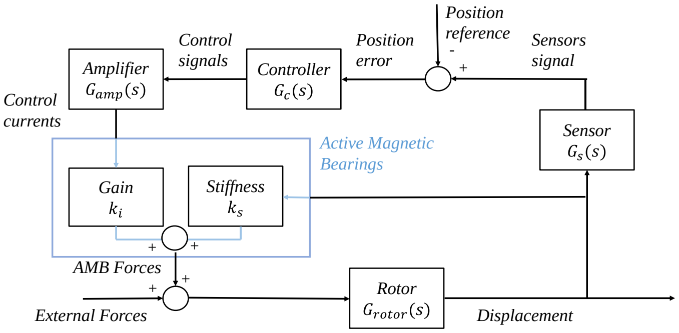


Figure 2. Block diagram of an AMB closed loop system.

3.1. AMB Closed Loop System Modeling

A state-space formulation was developed for each component to model the dynamics of the closed-loop system. The rotor radial state-space model was specifically constructed using a finite element method (FEM). This involved discretizing the rotor into N nodes along its geometric rotational axis. A Timoshenko modeling was utilized for the shafts, with a detailed description of the formulation available in [19] or [20]. The resulting equation that described the rotor dynamics is the following:

$$\begin{aligned} &M\ddot{\bar{q}} + \left(C + \Omega C_g\right)\dot{\bar{q}} + (K - K_s)\bar{q} = \bar{F}_{AMB} + \bar{F}_{ext} \\ &\bar{F}_{AMB} = K_i\bar{\imath} \end{aligned} \tag{2}$$

where M is the mass matrix, C is the damping matrix, C_g is the gyroscopic matrix, K is the stiffness matrix, Ω is the rotor rotational speed, $\bar{F}_{ext}$ are the external forces acting on the rotor, $\bar{F}_{AMB}$ are the AMB control forces, $\bar{q}$ is the vector that represents the position of every node of the rotor, $\bar{\imath}$ the vector of the control currents and, K_i and K_s respectively the matrix of current gains and the matrix of negative stiffnesses. Furthermore, a model order reduction technique was used on (2) to reduce the complexity of the system. In particular, modal truncation, described for example in [21], was used to eliminate all the rotor modes that are at a frequency above the range of interest for this type of application. Starting from (2), the rotor state space equation can be found:

$$\begin{aligned} &\dot{\bar{X}}_R = A_R\bar{X}_R + B_R\left(\bar{F}_{AMB} + \bar{F}_{ext}\right) \\ &\bar{q} = C_R\bar{X}_R \end{aligned} \tag{3}$$

where

$$\bar{X}_R = \begin{bmatrix} \bar{q} \\ \dot{\bar{q}} \end{bmatrix}, \quad A_R = \begin{bmatrix} [0] & I \\ -M^{-1}(K - K_s) & -M^{-1}\left(C + \Omega C_g\right) \end{bmatrix},$$

$$\mathrm{B_R} = \begin{bmatrix} [0] \\ M^{-1} \end{bmatrix}, \quad C_R = [I \quad [0]].$$

Moreover, linear state-space models were used to describe the other components of the AMB system, considering their unique characteristics and specifications. Combining the sensors, controller and actuators models, a second state-space system was found:

$$\begin{aligned} \dot{\bar{X}}_B &= A_B\bar{X}_B + B_B\bar{q} \\ \bar{F}_{AMB} &= C_B\bar{X}_B \end{aligned} \tag{4}$$

where $\bar{X}_B$ is the state and (A_B, B_B, C_B) are the matrices of the second state space model. Combining the equations (4) and (5), the state space model of the whole closed loop system was found:

$$\frac{d}{dt}\begin{bmatrix}\bar{X}_R \\ \bar{X}_B\end{bmatrix} = \begin{bmatrix} A_R & B_R C_B \\ B_B C_R & A_B \end{bmatrix}\begin{bmatrix}\bar{X}_R \\ \bar{X}_B\end{bmatrix} + \begin{bmatrix} B_R \\ [0] \end{bmatrix}\bar{F}_{ext} \tag{5}$$

The modeling of the whole system allows for simulating the dynamic behavior of the whole system. The main contributions of the term $\bar{F}_{ext}$ are the unbalance forces. Unbalance forces are synchronous forces with the speed of the rotor that are induced by the presence of unbalanced masses with respect to the axis of rotation of the rotor. The unbalance masses are due to inevitable manufacturing errors or rotor wear, and they are characteristic of a particular turbomachine. For a given axis x, at a fixed rotor speed Ω the unbalance force assumes the form:

$$F_{un_x} = \Omega^2 U\cos(\Omega t + \phi), \tag{6}$$

where ϕ is the phase of the unbalance respect other axis and U is the unbalance magnitude in kgm. In order to exactly reproduce the dynamic model, the machine is experimental identified to accurately assess the unbalance and to fine tune the state-space model (3). Some identification methods for AMB systems can be found in [22] or in [23].

3. Fault Dictionary

Due to the mechatronic structure of an Active Magnetic Bearing (AMB) system, failures can manifest in various forms, they can be software, electrical, or mechanical. These faults can result in the high-speed rotor making contact with its housing, potentially compromising the safety of the entire plant. Although touch-down bearings are designed to prevent direct contact between the rotor and housing, such an event must be avoided at all costs. To address the diverse range of faults that can arise in AMB systems, several measures can be taken, including redundancies, quality control, individual measures, and various control strategies. Active fault diagnostics and corrections can also be employed. Guidelines for designing a reliable system are provided in ISO 14839 [24] and API 617 [25] for turbomachinery on AMBs in the oil and gas field. A three-stage process for dealing with faults is described in [26], which involves determining the timing of the fault, identifying the faulty component, and identifying the type of fault. Given the complexity of the system, there are numerous reasons why malfunctions can occur, which can have different degrees of impact on system performance. [27] summarizes the main causes of malfunctions, their occurrence, and severity. This work takes into consideration the most common electrical soft faults, which can be modeled in three different ways: multiplicative, bias, and noise, as discussed in [28]. More specifically, this work considered the most frequent scenario, i.e., that of a single fault at a time. Figure 3 provides a summary of these faults.

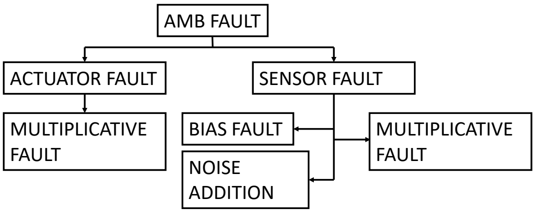


Figure 3. Resume of common electrical faults of AMB system

Steady-state signals, obtained from sensors and control systems are exploited to build images that serve as fault signatures. In particular, in this work the available considered signals were position signals from sensors, control signals from the controller, and current signals from the actuators, that are generally easily obtainable in an AMB system. Figure 4 summarizes the electrical signals that were chosen to build the dictionary. Since all signals exhibit periodic behavior, during normal machine operations at a constant fixed rotational speed, it is possible to create groups of images by representing the signals related to one of the two orthogonal control axes as functions of those related to the other axis, disregarding time as an independent variable and obtaining generalized orbits of the signals related to each bearing, as described in Figure 4.

The fault features were obtained by using a simulation tool capable of automatically building the entire fault dictionary once the fault conditions are modeled. The simulation tool is implemented in the Matlab-Simulink environment and allows for performing Monte Carlo simulations, by varying a selected set of component parameters using specific probability distributions or by adding noisy signals. The developed simulation tool was validated by comparing the results obtained with a commercial software (MADYN 2000).

Reference images, relative to non-faulty condition, are formed by simulating machine operations at a constant rotor speed and collecting signals while varying the component parameters within their tolerance ranges and accounting for normal levels of noise. The dictionary is completed by simulating the different faults that belong to the previously introduced fault classes. In detail, examples of each single soft fault signatures in the fault classes listed above, are obtained by randomly varying all the electrical parameters (gains, sensitivity and biases) within the tolerance ranges, but for the faulty component which varies outside this range, or by injecting noise with an abnormal standard deviation.

The variations of the component parameters in the tolerance ranges determine regions of confidence in the reference images relative to non-faulty condition where the generalized orbits should remain under fault-free conditions. If the orbits go beyond these regions, a fault has occurred. The built fault dictionary consists of images containing the deformed orbits with respect to the reference ones, given by simulating the different faults for the particular plant.

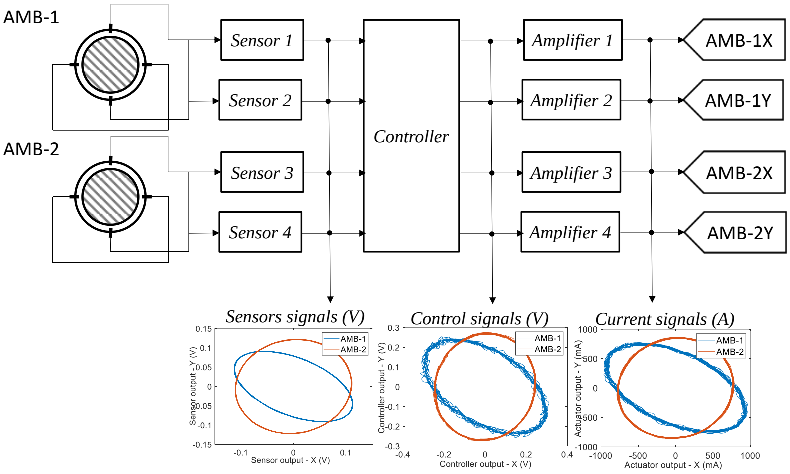


Figure 4. Summary of electrical signals that were chosen to build the dictionary.

Once the simulations are done, the feature images can be built. The time signals were normalized, and the orbits triplets were represented by images with a fixed resolution. Black and white images were generated through the Matplotlib library in Python, one for each orbit typology, one for the control signals, one for the position signals and the last for current signals. In the end, these images were concatenated into a unique RGB image, so that each RGB channel concerns only one orbit typology. In other words, for every triplet of images, related to a particular scenario, a unique RGB color was associated at each orbit typology, the red for the position signals, the blue for control signals and the green for current signals. Directly using images from available signals to train the classifier improves the ability of the system to understand the AMB system state, which offers a wider view and enhances the precision of information captured.

4. Classification Algorithm: Convolutional Neural Network

The proposed classifier exploits a Convolutional Neural Network (CNN), a neural network specialized for processing data that has a known, grid-like topology [29]. Examples include time-series data, which can be thought of as a 1D grid, and image data, which are 2D grid of pixels. LeCun et al [30] first applied the backpropagation algorithm to CNNs, with LeNet, a network developed to recognize handwritten digits. AlexNet [31] became the first modern deep convolutional neural network, representing a breakthrough in image classification.

A CNN is an architecture composed of multiple layers that collaborate to process and extract meaningful features from input data, each of them acting a different function in the network operations.

Convolutional layers consist of multiple filters, or kernels, that slide over the input data and perform convolutions. Each filter extracts specific features by detecting patterns and spatial information. The output of each filter is a feature map. An activation function, which introduces non-linearity into the network, is usually applied after each convolutional layer. Common choices include Rectified Linear Unit (ReLU), or variants such as Leaky ReLU or Parametric ReLU.

Pooling layers are frequently used after the convolutional layers to reduce the dimensionality of the feature maps and spatial dimensions. This down-sampling process helps to summarize and extract the most important information from the feature maps while reducing the computational complexity of the network. In other words, max pooling is a commonly used technique where the maximum value within a pooling window is selected as the representative value for that region.

After the convolutional and pooling layers, the feature maps are flattened into a one-dimensional vector. This vector is then fed into fully connected layers, also called dense layers, responsible for making predictions or classifications based on the extracted features. Like convolutional layers, activation functions are applied in fully connected layer to introduce non-linearity.

Finally, the last layer of CNNs is the output layer, which computes the network predictions. This time, the activation function used depends on the type of problem being solved. For classification tasks, as for this work, Softmax activation is commonly used to generate a probability distribution over the classes [32].

A loss function is used to measure the discrepancy between the predicted outputs and the ground truth labels. The choice of loss function depends on the problem type, such as categorical cross-entropy for multi-class classification, as in the proposed model [33].

During training, the parameters of the CNN are adjusted to minimize the loss function using optimization algorithms like stochastic gradient descent (SGD) or its variants, such Adam Optimizer [34]. This process, known as backpropagation, updates the weights and biases of the network to improve its performance. The structure of a CNN depicted in Figure 5, is very complex, however, many modern machine learning frameworks, like PyTorch, implement the above-described CNN operation.

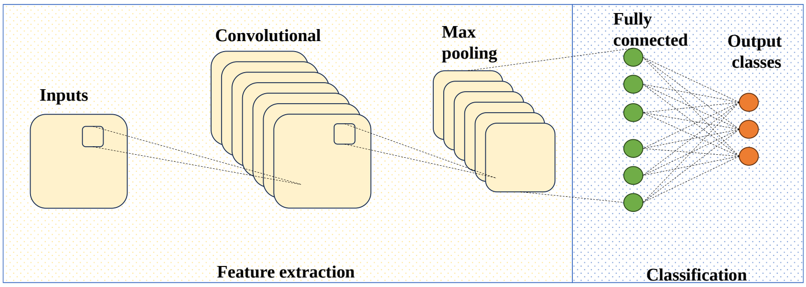


Figure 5. Schematic structure of CNN.

4.1 Proposed model

In this work, the CNN classifier has the final objective of recognizing 17 fault classes, among which 16 are fault classes and 1 is the nominal class related to the normal operation condition. These classes are summarized in Figure 6 and refer to Figure 3.

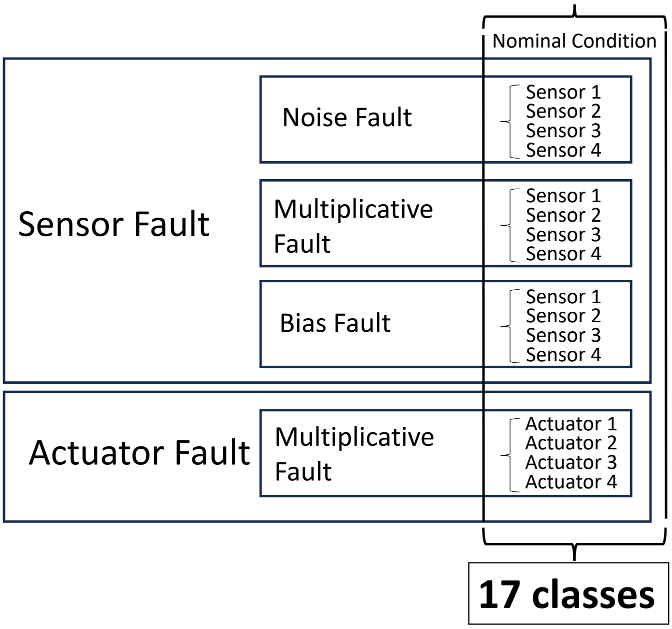


Figure 6. Summary of the classes to be classified.

The proposed CNN model consists of a series of alternately stacked convolutional, pooling and fully connected layers, as shown in Figure 7.

It employs convolutional layers to extract features, max pooling to down-sample the feature maps, and fully connected layers to map the extracted features to the output classes.

The first layer was a convolutional 2D, which has the images of orbit triplets as input channels while the output channels were the 17 classes described in Table 1. In this way, the layer was able to expand the information of the provided data. With the same purpose, a second convolutional layer was implemented. Then, a max pooling was computed, halving the dimensions of the images both for considering the expanded information and for preventing the gradient explosion. At this point, another convolutional 2D layer increased the dimension again, from 16 to 36 followed by another max pooling, for the same reasons explained above.

Subsequently, a batch normalization layer was applied to the extracted features in the previous layers, that allows for using high learning rates and for being quite unaffected by initialization. Moreover, this layer eliminates the need for dropout since it performs a regularization on the parameters, as explained in [35].

Afterwards, two additional convolutional layers were introduced. Unlike the preceding layers, these new layers aimed to combine or "zip" the output from the previous layer, thereby extracting the most significant features. This process helps to enhance the representation of the data by capturing more complex patterns and relationships.

The convolutional part of the network was concluded with a final max pooling operation. This step was followed by the transition to a fully connected layer. To prepare the data for this transition, the three-dimensional tensor resulting from the previous layers was flattened into a one-dimensional vector. Subsequently, three linear layers were trained in the network. Each of these layers was responsible for transforming the data and adjusting its dimensions to align with the output classes. In this case, the output dimension of each linear layer was rounded off to 17, which matches the number of classes in the classification task.

In the proposed model, each convolutional layer has a kernel dimension of 3x3 and a step dimension of 1x1, while the max pooling layers have a step dimension of 2x2. At least, each convolutional and fully connected layer, except the last one, was adhered with a ReLu activation function. Adam optimizer was used, which is computationally efficient, has low memory requirements, is invariant to diagonal rescaling of gradients, and is well suited to problems with large data and/or parameters [34]. The architecture was implemented using the PyTorch libraries.

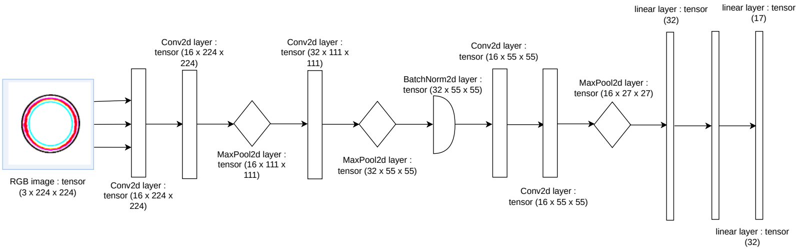

RGB image : tensor (3 x 224 x 224)
Conv2d layer : tensor (16 x 224 x 224)
Conv2d layer : tensor (16 x 224 x 224)
MaxPool2d layer : tensor (16 x 111 x 111)
Conv2d layer : tensor (32 x 111 x 111)
MaxPool2d layer : tensor (32 x 55 x 55)
BatchNorm2d layer : tensor (32 x 55 x 55)
Conv2d layer : tensor (16 x 55 x 55)
Conv2d layer : tensor (16 x 55 x 55)
MaxPool2d layer : tensor (16 x 27 x 27)
linear layer : tensor (32)
linear layer : tensor (32)
linear layer : tensor (17)

Figure 7. The proposed model structure.

5. Case Study

As a case study to assess the performance of the proposed diagnosis system an AMB supported system was considered. The case study is a real medium-size compressor supported by AMBs for the oil and gas field. Figure 8 illustrates the finite element model of the rotor under investigation with a mass of about 810 kg and a length of about 1.80 meters. Regarding the electrical part, the controller structure was an augmented PID. It has a decentralized structure, that divides the system into control axes and every control axis is controlled independently. For what concerns the other components, the system comprises switching pulse width modulation (PWM) amplifiers with heteropolar AMB actuators and inductive position sensors with a band of 4 kHz.

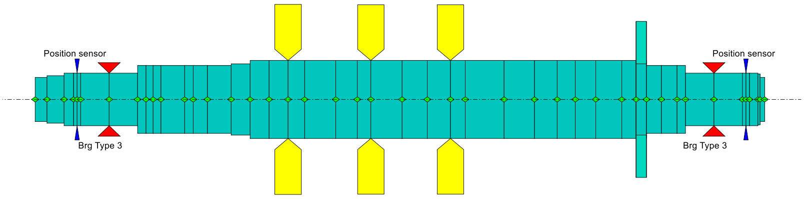


Figure 8. This image shows the finite element model of the rotor, where the red triangles are the bearings, the yellow elements are the disks, and the blue elements are the sensors.

Under nominal conditions, the system features a constant rotor rotation speed of 7800 rpm and an unbalance of $2.10 \times 10^{-3}\ kg \times m$ positioned at the center of mass of the rotor. The fault free condition is given by variations of the sensor sensitivities within a 2% tolerance of sensor, bias below 1 µm, and in the presence of a Gaussian white noise with a standard deviation of 1 µm. As for the actuators, a 2% tolerance of the DC gain is taken into consideration. Figure 9 displays an example of nominal condition orbits related to the fault-free signatures for the two radial bearings of the rotor.

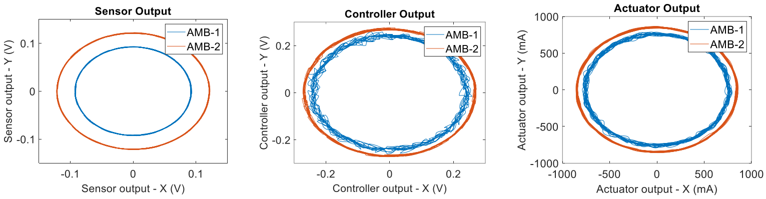


Figure 9. This image shows an example of nominal condition orbits. On the left the sensor output, in the center the controller output and on the right the actuator output are reported.

The faulty conditions that were taken into account to form the classes of the fault dictionary were computed through Monte Carlo analysis using specific parameter distribution. In particular, for each actuator multiplicative fault, the faulty actuator gain was chosen as a uniformly distributed random number out of the tolerance range centered in the nominal value in an interval ±50% the nominal value. Similarly, for each sensor multiplicative fault, the faulty sensor gain was chosen as a uniformly distributed random number out of the tolerance range centered in the nominal value in an interval ±50% of the nominal value. Instead, for each sensor noise fault, the faulty sensor is subjected to a gaussian additive noise with a standard deviation up to five times the nominal level of noise. Finally, for each sensor bias fault, the faulty sensor has random bias amplitude larger up to five times the nominal level of bias. Figure 10 shows an example of how orbits change shape when a specific fault occurs with respect to the orbits related to the nominal conditions reported for example in Figure 9.

The dataset used in the case study consists of 37600 RGB images obtained from different simulations, as described above in Section 3. More specifically, each image has a size of 3×224×224, where 3 represents the RGB channels and 224 x 224 is the number of pixels used for each image. Figure 11 gives an example of the used RGB images. The dataset is composed of 5600 examples of the fault free condition and 2000 examples of all the other 16 fault classes.

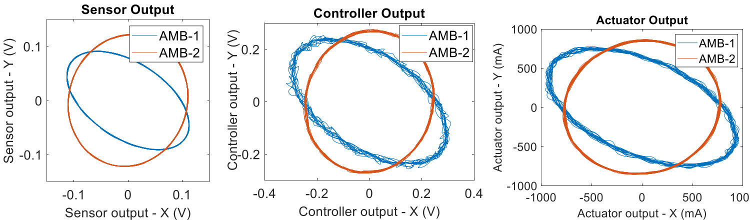


Figure 10. Example of signals orbits related to a sensor multiplicative fault, in particular, an AMB-1 x-axis multiplicative fault of -50% with respect to the nominal value.

Before training, data were pre-processed with aim of enhancing the model performance in terms of accuracy while also reducing noise interference. The dataset was divided into a training set, which includes 90% of the examples, and a test and a validation set, each which collects 5 % of the remaining examples.

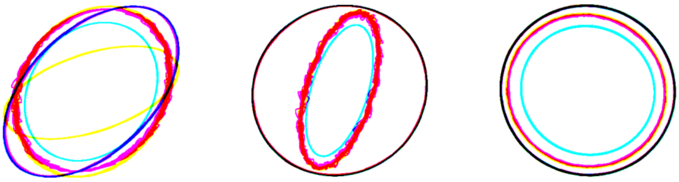


Figure 11. The RGB images used for training. In the right and central panel are shown, respectively, actuator and sensor gain fault. In the panel of left, nominal condition is shown.

6. Results and Discussion

The model was trained using the pre-processed training set for a total of 1000 epochs. Using the test set, the proposed CNN was validated, in fact, the reached test set accuracy was of about 93 %, while the training set accuracy reached was about 95 %. These results are presented in Figure 12, with the test set accuracy depicted in orange and the training set performance in pink.

Figure 13 shows the test and accuracy ratio, it represents the ratio between the accuracy of train and test reported in Figure 12. The descending trend observed in this plot suggests that the CNN's estimated parameters are not affected by overfitting. This indicates that the model has achieved a good level of generalization, meaning it can effectively generalize its learned patterns and make accurate predictions on unseen data.

Additionally, in Figure 14, the obtained confusion matrix is reported, and Table 1 presents the codebook of the labels. The horizontal axis of the matrix represents the actual faults, while the vertical axis represents the predicted faults. The diagonal elements show the percentage of correct predictions, the values above the diagonal indicate false positives, while the lower ones represent false negatives.

As shown by Figure 14, the confusion matrix is quite diagonal even though the lowest values are about 70 %.

It was found that the most part of misclassifications are related to soft faults related to small parametric deviations (with respect to the fault-free condition).

The achieved accuracy is really satisfactory for soft fault recognition taking into account also small parametric variations. This fact was due to two reasons. The first is that the regions of parameters variation related to faults border on the tolerance regions. The second reason is that the image resolution is limited. These facts lead some classes not to be recognized correctly, more precisely, the faulty class that are related to orbits that differ of a small amount with respect the reference ones in presence of small variation of a parameter out of the tolerance. This is a problem intrinsic in the definition of soft fault itself. No net borders between the faulty and the fault free condition exist, but in any case, positive or negative false related to these borderline situations have no severe consequences.

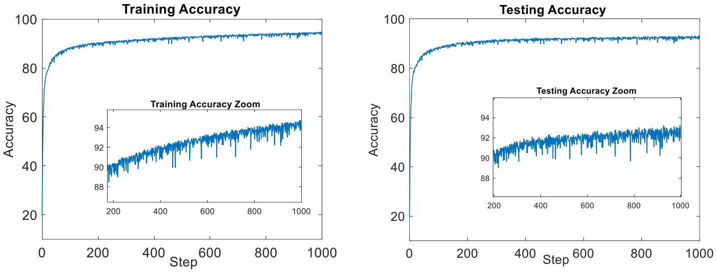


Figure 12. On the left plot the training accuracy for each epoch is shown, on the right the testing accuracy for each epoch is reported.

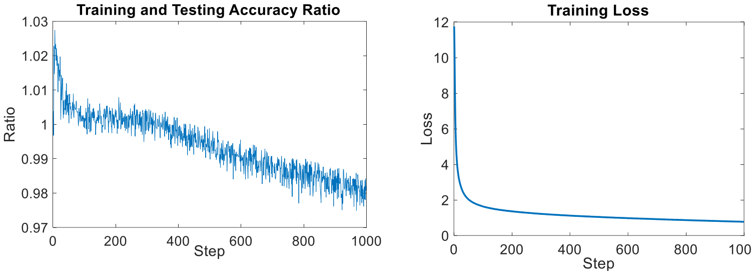
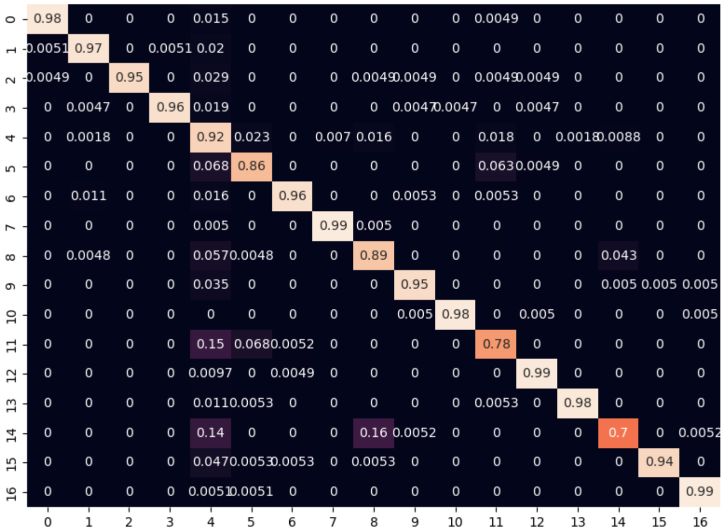


Figure 13. On the left the training and testing accuracy ratio plot is shown and on the right the training loss plot is reported.



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0	0.98	0	0	0	0.015	0	0	0	0	0	0	0.0049	0	0	0	0	0
1	0.0051	0.97	0	0.0051	0.02	0	0	0	0	0	0	0	0	0	0	0	0
2	0.0049	0	0.95	0	0.029	0	0	0	0.0049	0.0049	0	0.0049	0.0049	0	0	0	0
3	0	0.0047	0	0.96	0.019	0	0	0	0	0.0047	0.0047	0	0.0047	0	0	0	0
4	0	0.0018	0	0	0.92	0.023	0	0.007	0.016	0	0	0.018	0	0.0018	0.0088	0	0
5	0	0	0	0	0.068	0.86	0	0	0	0	0	0.063	0.0049	0	0	0	0
6	0	0.011	0	0	0.016	0	0.96	0	0	0.0053	0	0.0053	0	0	0	0	0
7	0	0	0	0	0.005	0	0	0.99	0.005	0	0	0	0	0	0	0	0
8	0	0.0048	0	0	0.057	0.0048	0	0	0.89	0	0	0	0	0	0.043	0	0
9	0	0	0	0	0.035	0	0	0	0	0.95	0	0	0	0	0.005	0.005	0.005
10	0	0	0	0	0	0	0	0	0	0.005	0.98	0	0.005	0	0	0	0.005
11	0	0	0	0	0.15	0.068	0.0052	0	0	0	0	0.78	0	0	0	0	0
12	0	0	0	0	0.0097	0	0.0049	0	0	0	0	0	0.99	0	0	0	0
13	0	0	0	0	0.011	0.0053	0	0	0	0	0	0.0053	0	0.98	0	0	0
14	0	0	0	0	0.14	0	0	0	0.16	0.0052	0	0	0	0	0.7	0	0.0052
15	0	0	0	0	0.047	0.0053	0.0053	0	0.0053	0	0	0	0	0	0	0.94	0
16	0	0	0	0	0.0051	0.0051	0	0	0	0	0	0	0	0	0	0	0.99

Figure 14. Obtained confusion matrix.

ID	Component	Kind of fault	ID	Component	Kind of fault
0	Actuator 1	Gain Fault	9	Sensor 2	Gain Fault
1	Actuator 2	Gain Fault	10	Sensor 2	Noise Fault
2	Actuator 3	Gain Fault	11	Sensor 3	Noise Fault
3	Actuator 4	Gain Fault	12	Sensor 3	Gain Fault
4	-	Nominal Condition	13	Sensor 3	Noise Fault
5	Sensor 1	Bias Fault	14	Sensor 4	Bias Fault
6	Sensor 1	Gain Fault	15	Sensor 4	Gain Fault
7	Sensor 1	Noise Fault	16	Sensor 4	Noise Fault
8	Sensor 2	Bias Fault			

Table 1. Codebook of the confusion matrix.

6. Conclusion

Active Magnetic Bearings are gaining popularity in a range of rotating machinery applications due to their high performance and to the elimination of cumbersome lubrication systems. On the other hand, to operate they need a complex closed loop mechatronic system. So, to ensure safe and reliable operation, fault detection and diagnostic systems are necessary. This proposed novel approach for fault diagnosing utilizes images of electrical signals available in an AMB system to train a classifier, a simple convolutional neural network, that was trained for detecting the most common soft electrical fault. Remarkably, the proposed classifier exhibits high accuracy and generalizability without requiring an extensive amount of data. This approach can be easily extended to other fault typologies and to other AMB supported systems.

Acknowledgments: This study was supported in part by Baker Hughes Company.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Srinivas R. Siva, Tiwari R., Kannababu Ch. Application of active magnetic bearings in flexible rotordynamic systems–A state-of-the-art review. *Mechanical Systems and Signal Processing*, **2018**, 106, 537-572.
2. Bleuler H., Cole M., Keogh P., Larsonneur R., Maslen E., Okada Y., Schweitzer G., A. Traxler. *Magnetic bearings: theory, design, and application to rotating machinery.* Springer Science & Business Media, 2009.
3. Tantau Mathias, Morantz Paul, Shore Paul. Position sensor for active magnetic bearing with commercial linear optical encoders. *CIRP Annals*, **2021**, 70.1, 419-422.
4. M. Basso, G. Donati and M. Mugnaini. A simulation tool for sensor selection in AMB rotor supported systems. In Proceedings of 2023 IEEE International Instrumentation and Measurement Technology Conference (I2MTC), Kuala Lumpur, Malaysia, 2023.
5. Da Silva, Gilberto Machado, Pederiva Robson. Fault diagnosis of active magnetic bearings. *Mechatronics*, **2022**, 84, 102801.
6. Sarmah Nilakshi, Tiwari Rajiv. Analysis and identification of the additive and multiplicative fault parameters in a cracked-bowed-unbalanced rotor system integrated with an auxiliary active magnetic bearing. *Mechanism and Machine Theory*, **2020**, 146, 103744.
7. Tsai Nan-Chyuan, King Yueh-Hsun, Lee Rong-Mao. Fault diagnosis for magnetic bearing systems. *Mechanical systems and signal Processing*, **2009**, 23.4, 1339-1351.
8. Caciotta M., Cerqua V., Leccese F., Giarnetti S., De Francesco E., De Francesco E.; Scaldarella N. A first study on prognostic system for electric engines based on Envelope Analysis. In Proceedings of 2014 IEEE Metrology for Aerospace (MetroAeroSpace), Benevento, Italy, 2014, 362-366.
9. Yan X., Zhang C. A., Liu Y. Multi-branch convolutional neural network with generalized shaft orbit for fault diagnosis of active magnetic bearing-rotor system. *Measurement*, **2021**, 171, 108778.

10. Jing L., Zhao M., Li P., Xu X. A convolutional neural network-based feature learning and fault diagnosis method for the condition monitoring of gearbox. *Measurement*, **2017**, 111, 1–10.
11. M. Basso, G. Donati and M. Mugnaini. Smart Fault Dictionary for Active Magnetic Bearings Systems. In Proceedings of International Workshop on Metrology for Industry 4.0&IoT (MetroInd4.0&IoT), Brescia, Italy, 2023.
12. Yan X., Sun Z., Zhao J., Shi Z., Zhang C. A. Fault diagnosis of active magnetic bearing–rotor system via vibration images. *Sensors*, **2019**, 19.2, 244.
13. Ciregan D., Meier U., Schmidhuber J., Schmidhuber Jürgen. Multi-column deep neural networks for image classification. In Proceedings of 2012 IEEE conference on computer vision and pattern recognition. IEEE, 2012, 3642-3649.
14. Ajit A., Acharya K., Samanta A. A review of convolutional neural networks. In Proceedings of 2020 international conference on emerging trends in information technology and engineering (ic-ETITE) IEEE, 2020, 1-5.
15. Fort A., Bertocci F., Mugnaini M., Vignoli, V., Gaggii V., Galasso A., Pieralli M. Availability modeling of a safe communication system for rolling stock applications. In Conference Record - IEEE Instrumentation and Measurement Technology Conference, 2013, pp. 427–430, 6555453.
16. Ceschini G., Mugnaini M., Masi A. A reliability study for a submarine compression application. *Microelectronics Reliability*, **2002**, 42(9-11), pp. 1377–1380.
17. Catelani M., Ciani L., Mugnaini M., Scarano V., Singuaroli R. Definition of safety levels and performances of safety: Applications for an electronic equipment used on rolling stock. In 2007 IEEE Instrumentation & Measurement Technology Conference IMTC 2007 (pp. 1-4).
18. Swanson E. E., Maslen E. H., Li G, Cloud C. H. Rotordynamic design audits of AMB supported machinery. In Proceedings of the 37th Turbomachinery Symposium. Texas A&M University. Turbomachinery Laboratories, 2008.
19. Lalanne M., Ferraris, G. *Rotordynamics prediction in engineering*. Wiley, 1998.
20. Friswell M. I., Penny J. E., Garvey S. D., Lees A. W. *Dynamics of rotating machines*. Cambridge university press, 2010.
21. Z. Gosiewski, Z. Kulesza. Virtual collocation of sensors and actuators for a flexible rotor supported by active magnetic bearings. In Proceedings of the 14th International Carpathian Control Conference (ICCC), Rytro, Poland, 2013, 94-99.
22. Aenis M., Knopf E., Nordmann R. Active magnetic bearings for the identification and fault diagnosis in turbomachinery. *Mechatronics*, **2002**, 12.8, 1011-1021.
23. Sun Z., He Y., Zhao J., Shi Z., Zhao L., Yu S. Identification of active magnetic bearing system with a flexible rotor. *Mechanical Systems and Signal Processing*, **2014**, 49.1-2, 302-316.
24. ISO 14839-3:2006 Mechanical vibration — Vibration of rotating machinery equipped with active magnetic bearings — Part 3: Evaluation of stability margin.
25. API Standard 617. Axial and centrifugal compressors and expander-compressors for petroeluem, chemical and gas industry services. Eight edition, American Petroleum Institute, Washington D.C, **2014**.
26. Frank Paul M., Ding Steven X., Marcu Teodor. Model-based fault diagnosis in technical processes. *Transactions of the Institute of Measurement and Control*, **2000**, 22.1, 57-101.
27. Lijesh K. P., Hirani H. Failure Mode and Effect Analysis of Active Magnetic Bearings. *Tribology in Industry*, **2016**, 38.1.
28. Kim Seung-Jong, Lee Chong-Won. Diagnosis of sensor faults in active magnetic bearing system equipped with built-in force transducers. *IEEE/ASME transactions on mechatronics*, **1999**, 4.2, 180-186.
29. I. Goodfellow, Y. Bengio, A. Courville. *Deep learning*. The MIT Press, 2016.
30. Y. Lecun, L. Bottou, Y. Bengio, P. Haffner. Gradient-based learning applied to document recognition. In Proceedings of the IEEE, vol. 86, no. 11, 2278-2324, Nov. 1998.
31. Krizhevsky Alex, Ilya Sutskever, Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60.6, **2017**, 84-90.
32. Liu W., Wen Y., Yu Z., Yang M. Large margin softmax loss for convolutional neural networks. *arXiv preprint arXiv*, **2016**, *1612.02295*.
33. Zhang Z., Sabuncu M. Generalized cross entropy loss for training deep neural networks with noisy labels. *Advances in neural information processing systems*, **2018**, 31.
34. Kingma D. P., Ba J. Adam: A Method for Stochastic Optimization. *ArXiv. /abs/1412.698*, **2014**.
35. Ioffe S., Szegedy C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *ArXiv. /abs/1502.03167*, **2015**.

Bibliography

[1] A. A. Abba Ari, O. K. Ngangmo, C. Titouna, O. Thiare, A. Mohamadou, and A. M. Gueroui, “Enabling privacy and security in cloud of things: Architecture, applications, security & privacy challenges,” *Applied Computing and Informatics*, vol. 20, no. 1/2, pp. 119–141, 2024.

[2] M. Abdelaty, S. Scott-Hayward, R. Doriguzzi-Corin, and D. Siracusa, “Gadot: Gan-based adversarial training for robust ddos attack detection,” in *2021 IEEE Conference on Communications and Network Security (CNS)*. IEEE, 2021, pp. 119–127.

[3] F. Adelantado *et al.*, “Understanding the limits of lorawan,” *IEEE Communications Magazine*, vol. 55, no. 9, pp. 34–40, 2017.

[4] M. Ahmid, O. Kazar, and E. Barka, “Internet of things overview: Architecture, technologies, application, and challenges,” in *Decision Making and Security Risk Management for IoT Environments*. Springer, 2024, pp. 1–19.

[5] A. Ajit, K. Acharya, and A. Samanta, “A review of convolutional neural networks,” in *Proceedings of the 2020 International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE)*. IEEE, 2020, pp. 1–5.

[6] A. S. D. Alluhaidan and P. Prabu, “End-to-end encryption in resource-constrained iot device,” *IEEE Access*, vol. 11, pp. 70 040–70 051, 2023.

[7] A. A. Almazroi and N. Ayub, “Deep learning hybridization for improved malware detection in smart internet of things,” *Scientific Reports*, vol. 14, no. 1, p. 7838, 2024.

[8] L. Alzubaidi, J. Bai, A. Al-Sabaawi, J. Santamaría, A. S. Albahri, B. S. N. Al-dabbagh, M. A. Fadhel, M. Manoufali, J. Zhang, A. H. Al-Timemy *et al.*, “A survey on deep learning tools dealing with data scarcity: definitions, challenges, solutions, tips, and applications,” *Journal of Big Data*, vol. 10, no. 1, p. 46, 2023.

[9] M. Ammar *et al.*, "Internet of things: A survey on the security of iot frameworks," *Journal of Information Security and Applications*, vol. 38, pp. 8–27, 2018.

[10] C. Andrade, "Sample size and its importance in research," *Indian journal of psychological medicine*, vol. 42, no. 1, pp. 102–103, 2020.

[11] M. Asam, S. H. Khan, A. Akbar, S. Bibi, T. Jamal, A. Khan, U. Ghafoor, and M. R. Bhutta, "Iot malware detection architecture using a novel channel boosted and squeezed cnn," *Scientific Reports*, vol. 12, no. 1, p. 15498, 2022.

[12] P. Asghari, A. M. Rahmani, and H. H. S. Javadi, "Service composition approaches in iot: A systematic review," *Journal of Network and Computer Applications*, vol. 120, pp. 61–77, 2018.

[13] Ö. A. Aslan and R. Samet, "A comprehensive review on malware detection approaches," *IEEE access*, vol. 8, pp. 6249–6271, 2020.

[14] https://aws.amazon.com/iot.

[15] C. Badii, P. Bellini, A. Difino, and P. Nesi, "Smart city iot platform respecting gdpr privacy and security aspects," *IEEE Access*, 2020.

[16] S. Bagchi, T. F. Abdelzaher, R. Govindan, P. Shenoy, A. Atrey, P. Ghosh, and R. Xu, "New frontiers in iot: Networking, systems, reliability, and security challenges," *IEEE Internet of Things Journal*, vol. 7, no. 12, pp. 11 330–11 346, 2020.

[17] M. Basso, G. Donati, and M. Mugnaini, "Smart fault dictionary for active magnetic bearings systems," in *2023 IEEE International Workshop on Metrology for Industry 4.0 & IoT (MetroInd4.0&IoT)*. IEEE, 2023, pp. 360–365.

[18] ——, "A simulation tool for sensor selection in amb rotor supported systems," in *2023 IEEE International Instrumentation and Measurement Technology Conference (I2MTC)*. IEEE, 2023, pp. 1–6.

[19] P. Bellini, D. Cenni *et al.*, "Smart city control room dashboards: Big data infrastructure, from data to decision support," in *Journal of Visual Languages and Computing*, 2020.

[20] P. Bellini, D. Nesi *et al.*, "Federation of smart city services via apis," in *Proc of 6th IEEE International Workshop on Sensors and Smart Cities, with IEEE SmartComp*, 2019.

[21] H. Bleuler, M. Cole, P. Keogh, R. Larsonneur, E. Maslen, Y. Okada, G. Schweitzer, and A. Traxler, *Magnetic bearings: theory, design, and application to rotating machinery.* Springer Science & Business Media, 2009.

[22] K. H. Brodersen, C. S. Ong, K. E. Stephan, and J. M. Buhmann, "The balanced accuracy and its posterior distribution," in *2010 20th International Conference on Pattern Recognition*, 2010, pp. 3121–3124.

[23] M. Caciotta, V. Cerqua, F. Leccese, S. Giarnetti, E. De Francesco, and N. Scaldarella, "A first study on prognostic system for electric engines based on envelope analysis," in *Proceedings of the 2014 IEEE Metrology for Aerospace (MetroAeroSpace)*, Benevento, Italy, 2014, pp. 362–366.

[24] M. Catelani, L. Ciani, M. Mugnaini *et al.*, "Definition of safety levels and performances of safety: Applications for an electronic equipment used on rolling stock," in *2007 IEEE Instrumentation & Measurement Technology Conference IMTC 2007*, 2007, pp. 1–4.

[25] G. Ceschini, M. Mugnaini, and A. Masi, "A reliability study for a submarine compression application," *Microelectronics Reliability*, vol. 42, no. 9-11, pp. 1377–1380, 2002.

[26] R. Chaganti, V. Ravi, and T. D. Pham, "A multi-view feature fusion approach for effective malware classification using deep learning," *Journal of information security and applications*, vol. 72, p. 103402, 2023.

[27] H. Chen and M. A. Babar, "Security for machine learning-based software systems: A survey of threats, practices, and challenges," *ACM Computing Surveys*, vol. 56, no. 6, pp. 1–38, 2024.

[28] B. Cheng, G. Solmaz *et al.*, "Fogflow: Easy programming of iot services over cloud and edges for smartcities," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 696–707, 2018.

[29] D. Ciregan, U. Meier, and J. Schmidhuber, "Multi-column deep neural networks for image classification," in *Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition.* IEEE, 2012, pp. 3642–3649.

[30] A. Čolaković and M. Hadžialić, "Internet of things (iot): A review of enabling technologies, challenges, and open research issues," *Computer networks*, vol. 144, pp. 17–39, 2018.

[31] J. W. Conard, "Services and protocols of the data link layer," *Proceedings of the IEEE*, vol. 71, no. 12, pp. 1378–1383, 1983.

[32] F.-A. Croitoru, V. Hondru, R. T. Ionescu, and M. Shah, "Diffusion models in vision: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 9, pp. 10 850–10 869, 2023.

[33] G. M. Da Silva and R. Pederiva, "Fault diagnosis of active magnetic bearings," *Mechatronics*, vol. 84, p. 102801, 2022.

[34] F. Deldar and M. Abadi, "Deep learning for zero-day malware detection and classification: A survey," *ACM Computing Surveys*, 2023.

[35] J. S.-P. Díaz and Á. L. García, “Study of the performance and scalability of federated learning for medical imaging with intermittent clients,” *Neurocomputing*, vol. 518, pp. 142–154, 2023.

[36] G. Donati, M. Basso, G. A. Manduzio, M. Mugnaini, T. Pecorella, and C. Camerota, “A convolutional neural network for electrical fault recognition in active magnetic bearing systems,” *Sensors*, vol. 23, no. 16, p. 7023, 2023.

[37] S. Dramé-Maigné, M. Laurent, L. Castillo, and H. Ganem, “Centralized, distributed, and everything in between: Reviewing access control solutions for the iot,” *ACM Computing Surveys (CSUR)*, vol. 54, no. 7, pp. 1–34, 2021.

[38] M. Eliasziw and A. Donner, “Application of the mcnemar test to non-independent matched pair data,” *Statistics in medicine*, vol. 10, no. 12, pp. 1981–1991, 1991.

[39] A. A. Elngar, R. Chowdhury, M. Elhoseny, and V. E. Balas, *Applications of Computational Intelligence in Multi-Disciplinary Research.* Academic Press, 2022.

[40] M. A. Ferrag, O. Friha, D. Hamouda, L. Maglaras, and H. Janicke, “Edge-iiotset: A new comprehensive realistic cyber security dataset of iot and iiot applications for centralized and federated learning,” *IEEE Access*, vol. 10, pp. 40 281–40 306, 2022.

[41] G. M. Foody, “Sample size determination for image classification accuracy assessment and comparison,” *International Journal of Remote Sensing*, vol. 30, no. 20, pp. 5273–5291, 2009.

[42] A. Fort, F. Bertocci, M. Mugnaini *et al.*, “Availability modeling of a safe communication system for rolling stock applications,” in *Conference Record - IEEE Instrumentation and Measurement Technology Conference*, 2013, pp. 427–430.

[43] G. Fortino, C. Savaglio, C. E. Palau, J. S. De Puga, M. Ganzha, M. Paprzycki, M. Montesinos, A. Liotta, and M. Llop, “Towards multi-layer interoperability of heterogeneous iot platforms: The inter-iot approach,” *Integration, interconnection, and interoperability of IoT systems*, pp. 199–232, 2018.

[44] P. M. Frank, S. X. Ding, and T. Marcu, “Model-based fault diagnosis in technical processes,” *Transactions of the Institute of Measurement and Control*, vol. 22, no. 1, pp. 57–101, 2000.

[45] G. Gardašević, M. Veletić, N. Maletić, D. Vasiljević, I. Radusinović, S. Tomović, and M. Radonjić, “The iot architectural framework, design issues and application domains,” *Wireless personal communications*, vol. 92, pp. 127–148, 2017.

[46] A. Gaurav, B. B. Gupta, and P. K. Panigrahi, “A comprehensive survey on machine learning approaches for malware detection in iot-based enterprise information system,” *Enterprise Information Systems*, vol. 17, no. 3, p. 2023764, 2023.

[47] A. Ghasempour, “Internet of things in smart grid: Architecture, applications, services, key technologies, and challenges,” *Inventions*, vol. 4, no. 1, p. 22, 2019.

[48] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning.* MIT Press, 2016.

[49] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial nets,” *Advances in neural information processing systems*, vol. 27, 2014.

[50] https://cloud.google.com/solutions/io.

[51] A. Gosiewska, A. Kozak, and P. Biecek, “Simpler is better: Lifting interpretability-performance trade-off via automated feature engineering,” *Decision Support Systems*, vol. 150, p. 113556, 2021.

[52] K. Grosse, N. Papernot, P. Manoharan, M. Backes, and P. McDaniel, “Adversarial examples for malware detection,” in *Computer Security–ESORICS 2017: 22nd European Symposium on Research in Computer Security, Oslo, Norway, September 11-15, 2017, Proceedings, Part II 22.* Springer, 2017, pp. 62–79.

[53] A. Gupta, R. Christie, and R. Manjula, “Scalability in internet of things: features, techniques and research challenges,” *Int. J. Comput. Intell. Res*, vol. 13, no. 7, pp. 1617–1627, 2017.

[54] M. K. Hasan, A. A. Habib, Z. Shukur, F. Ibrahim, S. Islam, and M. A. Razzaque, “Review on cyber-physical and cyber-security system in smart grid: Standards, protocols, constraints, and recommendations,” *Journal of Network and Computer Applications*, vol. 209, p. 103540, 2023.

[55] V. Hassija, V. Chamola, V. Saxena, D. Jain, P. Goyal, and B. Sikdar, “A survey on iot security: application areas, security threats, and solution architectures,” *IEEe Access*, vol. 7, pp. 82 721–82 743, 2019.

[56] J. Ho, A. Jain, and P. Abbeel, “Denoising diffusion probabilistic models,” *Advances in neural information processing systems*, vol. 33, pp. 6840–6851, 2020.

[57] ——, “Denoising diffusion probabilistic models,” 2020. [Online]. Available: https://arxiv.org/abs/2006.11239

[58] Y. Hu, O. W. Taha, and K. Yang, “Fault detection in active magnetic bearings using digital twin technology,” *Applied Sciences*, vol. 14, no. 4, p. 1384, 2024.

[59] F. Hussain, R. Hussain, S. A. Hassan, and E. Hossain, "Machine learning in iot security: Current solutions and future challenges," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 3, pp. 1686–1721, 2020.

[60] https://ww.ibm.com/watson.

[61] International Organization for Standardization and International Electrotechnical Commission, *ISO/IEC 20924:2024 Information technology - Vocabulary - Terms and definitions*, Geneva, 2024, international Standard.

[62] W. Issa, N. Moustafa, B. Turnbull, N. Sohrabi, and Z. Tari, "Blockchain-based federated learning for securing internet of things: A comprehensive survey," *ACM Computing Surveys*, vol. 55, no. 9, pp. 1–43, 2023.

[63] A. Jahangeer, S. U. Bazai, S. Aslam, S. Marjan, M. Anas, and S. H. Hashemi, "A review on the security of iot networks: From network layer's perspective," *IEEE Access*, vol. 11, pp. 71 073–71 087, 2023.

[64] J. Jeon, J. H. Park, and Y.-S. Jeong, "Dynamic analysis for iot malware detection with convolution neural network model," *IEEE Access*, vol. 8, pp. 96 899–96 911, 2020.

[65] X. Jiang, S. Liu, A. Gember-Jacobson, A. N. Bhagoji, P. Schmitt, F. Bronzino, and N. Feamster, "Netdiffusion: Network data augmentation through protocol-constrained traffic generation," *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, vol. 8, no. 1, feb 2024.

[66] L. Jing, M. Zhao, P. Li, and X. Xu, "A convolutional neural network-based feature learning and fault diagnosis method for the condition monitoring of gearbox," *Measurement*, vol. 111, pp. 1–10, 2017.

[67] V. Karagiannis, P. Chatzimisios, F. Vazquez-Gallego, and J. Alonso-Zarate, "A survey on application layer protocols for the internet of things," *Transaction on IoT and Cloud computing*, vol. 3, no. 1, pp. 11–17, 2015.

[68] S.-J. Kim and C.-W. Lee, "Diagnosis of sensor faults in active magnetic bearing system equipped with built-in force transducers," *IEEE/ASME Transactions on Mechatronics*, vol. 4, no. 2, pp. 180–186, 1999.

[69] V. R. Konduru and M. R. Bharamagoudra, "Challenges and solutions of interoperability on iot: How far have we come in resolving the iot interoperability issues," in *2017 International Conference On Smart Technologies For Smart Nation (SmartTechCon)*. IEEE, 2017, pp. 572–576.

[70] M. Lalanne and G. Ferraris, *Rotordynamics prediction in engineering.* Wiley, 1998.

[71] E. Lear, R. Droms, and D. Romascanu, "Manufacturer usage description specification," Request for Comments 8520, 2019. [Online]. Available: https://doi.org/10.17487/RFC8520

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

[73] A. L'heureux, K. Grolinger, H. F. Elyamany, and M. A. Capretz, "Machine learning with big data: Challenges and approaches," *Ieee Access*, vol. 5, pp. 7776–7797, 2017.

[74] K. P. Lijesh and H. Hirani, "Failure mode and effect analysis of active magnetic bearings," *Tribology in Industry*, vol. 38, no. 1, 2016.

[75] I. I. A. S. MANNER, "Industry 4.0 cybersecurity: Challenges & recommendations," *https://www.enisa.europa.eu/publications/industry-4-0-cybersecurity-challenges-and-recommendations*, 2019.

[76] K. Mekki *et al.*, "Overview of cellular lpwan technologies for iot deployment: Sigfox, lorawan, and nb-iot," in *2018 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*. IEEE, 2018.

[77] https://azure.microsoft.com/en-us/overview/iot.

[78] A. Nascita, F. Cerasuolo, D. Di Monda, J. T. A. Garcia, A. Montieri, and A. Pescapè, "Machine and deep learning approaches for iot attack classification," in *IEEE INFOCOM 2022-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 2022, pp. 1–6.

[79] L. Nataraj, V. Yegneswaran, P. Porras, and J. Zhang, "A comparative assessment of malware classification using binary texture analysis and dynamic analysis," in *Proceedings of the 4th ACM Workshop on Security and Artificial Intelligence*, ser. AISec '11. New York, NY, USA: Association for Computing Machinery, 2011, pp. 21–30.

[80] G. Nebbione and M. C. Calzarossa, "Security of iot application layer protocols: Challenges and findings," *Future Internet*, vol. 12, no. 3, p. 55, 2020.

[81] E. Oja, H. Ogawa, and J. Wangviwattana, "Principal component analysis by homogeneous neural networks, part¡ cd02d35. gif¿: The weighted subspace criterion," *IEICE Transactions on Information and Systems*, vol. 75, no. 3, pp. 366–375, 1992.

[82] A. Oliva and A. Torralba, "Modeling the shape of the scene: A holistic representation of the spatial envelope," *International journal of computer vision*, vol. 42, pp. 145–175, 2001.

[83] N. Pachhala, S. Jothilakshmi, and B. P. Battula, "A comprehensive survey on identification of malware types and malware classification using machine learning techniques," in *2021 2nd International Conference on Smart Electronics and Communication (ICOSEC)*. IEEE, 2021, pp. 1207–1214.

[84] K. K. Patel, S. M. Patel, and P. Scholar, "Internet of things-iot: definition, characteristics, architecture, enabling technologies, application & future challenges," *International journal of engineering science and computing*, vol. 6, no. 5, 2016.

[85] N. Patel and A. Singh, "Security issues, attacks and countermeasures in layered iot ecosystem." *International Journal of Next-Generation Computing*, vol. 14, no. 2, 2023.

[86] Q.-V. Pham, K. Dev, P. K. R. Maddikunta, T. R. Gadekallu, T. Huynh-The *et al.*, "Fusion of federated learning and industrial internet of things: A survey," *arXiv preprint arXiv:2101.00798*, 2021.

[87] G. P. Pinto, P. K. Donta, S. Dustdar, and C. Prazeres, "A systematic review on privacy-aware iot personal data stores," *Sensors*, vol. 24, no. 7, p. 2197, 2024.

[88] P. P. Ray, "A survey of iot cloud platforms," *Future Computing and Informatics Journal*, vol. 1, no. 1-2, pp. 35–46, 2016.

[89] V. Rey, P. M. S. Sánchez, A. H. Celdrán, and G. Bovet, "Federated learning for malware detection in iot devices," *Computer Networks*, vol. 204, p. 108693, 2022.

[90] P. Riehmann, M. Hanfler, and B. Froehlich, "Interactive sankey diagrams," in *IEEE Symposium on Information Visualization, 2005. INFOVIS 2005.* IEEE, 2005, pp. 233–240.

[91] J. Saleem, M. Hammoudeh, U. Raza, B. Adebisi, and R. Ande, "Iot standardisation: Challenges, perspectives and solution," in *Proceedings of the 2nd international conference on future networks and distributed systems*, 2018, pp. 1–9.

[92] N. Sarmah and R. Tiwari, "Analysis and identification of the additive and multiplicative fault parameters in a cracked-bowed-unbalanced rotor system integrated with an auxiliary active magnetic bearing," *Mechanism and Machine Theory*, vol. 146, p. 103744, 2020.

[93] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 618–626.

[94] H. Seo, J. Park, S. Oh, M. Bennis, and S.-L. Kim, "16 federated knowledge distillation," *Machine Learning and Wireless Communications*, p. 457, 2022.

[95] J. Sevilla, L. Heim, A. Ho, T. Besiroglu, M. Hobbhahn, and P. Villalobos, "Compute trends across three eras of machine learning," in *2022 International Joint Conference on Neural Networks (IJCNN).* IEEE, Jul. 2022.

[96] M. Shahrokh Esfahani and E. R. Dougherty, “Effect of separate sampling on classification accuracy,” *Bioinformatics*, vol. 30, no. 2, pp. 242–250, 2014.

[97] K. Shaukat, S. Luo, and V. Varadharajan, “A novel deep learning-based approach for malware detection,” *Engineering Applications of Artificial Intelligence*, vol. 122, p. 106030, 2023.

[98] K. Shmelkov, C. Schmid, and K. Alahari, “How good is my gan?” in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 213–229.

[99] C. Shorten and T. M. Khoshgoftaar, “A survey on image data augmentation for deep learning,” *Journal of big data*, vol. 6, no. 1, pp. 1–48, 2019.

[100] N. Siddique, S. Paheding, C. P. Elkin, and V. Devabhaktuni, “U-net and its variants for medical image segmentation: A review of theory and applications,” *IEEE access*, vol. 9, pp. 82 031–82 057, 2021.

[101] https://siemens.mindsphere.io/en.

[102] https://www.snap4city.org.

[103] H. R. Sofaer, J. A. Hoeting, and C. S. Jarnevich, “The area under the precision-recall curve as a performance metric for rare binary events,” *Methods in Ecology and Evolution*, vol. 10, no. 4, pp. 565–577, 2019.

[104] A. Souri and R. Hosseini, “A state-of-the-art survey of malware detection approaches using data mining techniques,” *Human-centric Computing and Information Sciences*, vol. 8, no. 1, pp. 1–22, 2018.

[105] R. S. Srinivas, R. Tiwari, and C. Kannababu, “Application of active magnetic bearings in flexible rotordynamic systems-a state-of-the-art review,” *Mechanical Systems and Signal Processing*, vol. 106, pp. 537–572, 2018.

[106] D. Stilinski and J. Owen, “Federated learning for secure and decentralized ai in the internet of things (iot),” *https://www.researchgate.net/publication/387538958*$_F$*ederated*$_L$*earning*$_f$*or*$_S$*ecure*$_a$*nd*$_D$*ecentrali*

[107] X. Su, J. Song, C. Meng, and S. Ermon, “Dual diffusion implicit bridges for image-to-image translation,” *arXiv preprint arXiv:2203.08382*, 2022.

[108] E. E. Swanson, E. H. Maslen, G. Li, and C. H. Cloud, “Rotordynamic design audits of amb supported machinery,” in *Proceedings of the 37th Turbomachinery Symposium*. Texas A&M University, Turbomachinery Laboratories, 2008.

[109] R. Taheri, M. Shojafar, M. Alazab, and R. Tafazolli, “Fed-iiot: A robust federated malware detection architecture in industrial iot,” *IEEE transactions on industrial informatics*, vol. 17, no. 12, pp. 8442–8452, 2020.

[110] M. Tantau, P. Morantz, and P. Shore, "Position sensor for active magnetic bearing with commercial linear optical encoders," *CIRP Annals*, vol. 70, no. 1, pp. 419–422, 2021.

[111] N. C. Tsai, Y. H. King, and R. M. Lee, "Fault diagnosis for magnetic bearing systems," *Mechanical Systems and Signal Processing*, vol. 23, no. 4, pp. 1339–1351, 2009.

[112] M. Umashankar, S. Mallikarjunaswamy, N. Sharmila, D. M. Kumar, and K. Nataraj, "A survey on iot protocol in real-time applications and its architectures," in *ICDSMLA 2021: Proceedings of the 3rd International Conference on Data Science, Machine Learning and Applications.* Springer, 2023, pp. 119–130.

[113] D. A. Van Dyk and X.-L. Meng, "The art of data augmentation," *Journal of Computational and Graphical Statistics*, vol. 10, no. 1, pp. 1–50, 2001.

[114] M. Venkatasubramanian, A. H. Lashkari, and S. Hakak, "Iot malware analysis using federated learning: A comprehensive survey," *IEEE Access*, vol. 11, pp. 5004–5018, 2023.

[115] ——, "Iot malware analysis using federated learning: A comprehensive survey," *IEEE Access*, 2023.

[116] G. Wang, X. Jia, J. Li, F. Li, Z. Liu, and B. Gong, "Current state and development of the research on solid particle erosion and repair of turbomachine blades," in *Re-engineering Manufacturing for Sustainability: Proceedings of the 20th CIRP International Conference on Life Cycle Engineering*, Singapore, 2013, pp. 633–638.

[117] N. Wang, P. Wang, A. Alipour-Fanid, L. Jiao, and K. Zeng, "Physical-layer security of 5g wireless networks for iot: Challenges and opportunities," *IEEE internet of things journal*, vol. 6, no. 5, pp. 8169–8181, 2019.

[118] W. Wang, M. Zhu, X. Zeng, X. Ye, Y. Sheng, and Y. Huang, "Hast-ids: Learning hierarchical spatial-temporal features using deep neural networks to improve intrusion detection," *IEEE Access*, vol. 6, pp. 1792–1806, 2017.

[119] X. Yan, Z. Sun, J. Zhao, Z. Shi, and C. A. Zhang, "Fault diagnosis of active magnetic bearing–rotor system via vibration images," *Sensors*, vol. 19, no. 2, p. 244, 2019.

[120] X. Yan, C. A. Zhang, and Y. Liu, "Multi-branch convolutional neural network with generalized shaft orbit for fault diagnosis of active magnetic bearing-rotor system," *Measurement*, vol. 171, p. 108778, 2021.

[121] W. Ye, G. Zheng, X. Cao, Y. Ma, X. Hu, and A. Zhang, "Spurious correlations in machine learning: A survey," *arXiv preprint arXiv:2402.12715*, 2024.

[122] X. Ying, "An overview of overfitting and its solutions," in *Journal of physics: Conference series*, vol. 1168. IOP Publishing, 2019, p. 022022.

[123] A. Yousefpour *et al.*, "All one needs to know about fog computing and related edge computing paradigms: A complete survey," *Journal of Systems Architecture*, vol. 98, pp. 289–330, 2019.

[124] J.-J. Zhu, M. Yang, and Z. J. Ren, "Machine learning in environmental research: common pitfalls and best practices," *Environmental Science & Technology*, vol. 57, no. 46, pp. 17 671–17 689, 2023.