WASHINGTON UNIVERSITY IN ST.LOUIS

School of Engineering & Applied Science
Department of Computer Science and Engineering

Dissertation Examination Committee:
Raj Jain, Chair
Roger Chamberlain
Chien-Ju Ho
Khaled Khan
Ning Zhang

Addressing Pragmatic Challenges in Utilizing AI for Security of Industrial IoT

by
Maede Zolanvari

A dissertation presented to
The Graduate School
of Washington University in
partial fulfillment of the
requirements for the degree
of Doctor of Philosophy

December 2021
St. Louis, Missouri

Table of Contents

List of Figures

List of Tables

Acknowledgments

I want to express my appreciation to all those who helped me throughout the pursuit of my Ph.D. degree, my family, friends, colleagues, professors, staff, and the university of Washington University in St. Louis.

I want to thank my adviser, Dr. Raj Jain, for his guidance and always motivating us to aim for higher goals. I am very grateful to my committee members, Dr. Roger Chanberlain, Dr.Chien-Ju Ho, Dr. Khaled Khan, and Dr. Ning Zhang for providing valuable feedback constructive suggestions throughout my research.

I would like to thank the funding agencies NSF and QNRF for supporting my research. This dissertation was made possible by NPRP10-0206-170360 and NSF grant CNS-1718929.

Last but not least, I want to extend my gratitude to my one and only, my husband, Josh Hoyt. He helped me keep my sanity during this tough journey. If not for his support over the last five years, I would have given up and would not be where I am today.

Maede Zolanvari

Washington University in Saint Louis

December 2021

Dedicated to those who fight hard in life and never give up.

ABSTRACT OF THE DISSERTATION

Addressing Pragmatic Challenges in Utilizing AI for Security of Industrial IoT

by

Maede Zolanvari

Doctor of Philosophy in Computer Science

Washington University in St. Louis, 2021

Professor Raj Jain, Chair

Industrial control systems (ICSs) are an essential part of every nation's critical infrastructure and have been utilized for a long time to supervise industrial machines and processes. Today's ICSs are substantially different from the information technology (IT) devices a decade ago. The integration of internet of things (IoT) technology has made them more efficient and optimized, improved automation, and increased quality and compliance. Now, they are a sub (and arguably the most critical) part of IoT's domain, called industrial IoT (IIoT).

In the past, to secure ICSs from malicious outside attack, these systems were isolated from the outside world. However, recent advances, increased connectivity with corporate networks, and utilization of internet communications to transmit the information more conveniently have introduced the possibility of cyber-attacks against these systems. Due to the sensitive nature of the industrial applications, security is the foremost concern.

We discuss why despite the exceptional performance of artificial intelligent (AI) and machine learning (ML), industry leaders still have a hard time utilizing these models in practice as a standalone units. The goal of this dissertation is to address some of these challenges to help pave the way of utilizing smarter and more modern security solutions in these systems.

To be specific, here, we focus on data scarcity for the AI, black-box nature of the AI, high computational load of the AI.

Industrial companies almost never release their network data, because they are obligated to follow confidentiality laws and user privacy restrictions. Hence, real-world IIoT datasets are not available for security research area, and we face a data scarcity challenge in IIoT security research community. In this domain, the researchers usually have to resort to commercial or public datasets that are not specific to this domain. In our work, we have developed a real-world testbed that resembles an actual industrial plant. We have emulated a popular industrial system in water treatment processes. So, we could collect datasets containing realist traffic to conduct our research.

There exists several specific characteristics of IIoT networks that are unique to them. We have provided an extensive study to figure out them and incorporate them in the design. We have gathered information on relevant cyber-attacks in IIoT systems to run them against the system to gather realistic datasets containing both normal and attack traffic analogous to real industrial network traffic. Their particular communication protocols are also their specific to them. We have implemented one of the most popular one in our dataset. Another attribute that distinguishes the security of these systems from others is the imbalanced data. The number of attack samples are significantly lower compared to the enormous number of normal traffic that flows in the system daily. We have made sure we build our datasets compliant with all the specific attributes of an IIoT.

Another challenge that we address here is the “black box" nature of learning models that creates hurdles in generating adequate trust in their decisions. Thus, they are seldom utilized as a standalone unit in IIoT high-risk applications. Explainable AI (XAI) has gained an increasing interest in recent years to help with this problem. However, most of the research

works that have been done so far focus on image applications or are very slow. For applications such as security of IIoT, we deal with numerical data and low latency is of utmost importance. In this dissertation, we propose a universal XAI model named Transparency Relying Upon Statistical Theory (TRUST). TRUST is model-agnostic, high-performing, and suitable for numerical applications. We prove its superiority compared to another popular XAI model in performance regarding speed and being able to successfully reason the AI's behavior.

When dealing with the IoT technology, especially industrial IoT, we deal with a massive amount of data streaming to and from the IoT devices. In addition, the availability and reliability constraints of industrial systems require them to operate at a fast pace and avoid creating any bottleneck in the system. High computational load of complex AI models might cause a burden by having to deal with a large number of data and producing the results not as fast as required. In this dissertation, we utilize distributed computing in the form of edge/cloud structure to address these problems. We propose Anomaly Detection using Distributed AI (ADDAI) that can easily span out geographically to cover a large number of IoT sources. Due to its distributed nature, it guarantees critical IIoT requirements such as high speed, robustness against a single point of failure, low communication overhead, privacy, and scalability. We formulate the communication cost which is minimized and the improvement in performance.

Chapter 1

Introduction

Internet of Things (IoT) is becoming an integral part of nowadays technological world. Industrial revolution has been playing a major role in the advancement of current technologies. Industrial Internet of Things (IIoT) is a reference to leveraging the IoT technology in Industrial Control Systems (ICS). ICSs are the foundations of the critical infrastructures of a nation. Industrial processes such as smart manufacturing, oil and gas exploration, water distribution and treatment, etc. rely heavily on these systems. Utilizing IoT technology in ICSs enhances network intelligence in the optimization and automation of industrial operations. However, as everything comes with a cost, these advancements have created new vulnerable surfaces for malicious threats against these systems. [1]

The legacy ICSs were isolated from the outside world. With no Internet connection, there was no need to be concerned about any outsider threat. Additionally, the whole platform was pre-defined and pre-programmed. Therefore, it was easy to detect any out-of-ordinary behavior

[1]It is important to note that, in this proposal, by "security" for these systems, we mean network security. Utilizing AI in other aspects of security, such as fraud detection, email security, dark data discovery, facial recognition to verify personal identities, etc. is outside the scope of this dissertation.

caused by an insider intruder or even a faulty part. Rule-based security measures would suffice and be adequate to guarantee smooth operations and protection against malicious intends.

Due to the isolated nature of ICSs, individual vendors often developed their own hardware, software and communication protocols. As a result, most of these products are proprietary and vendor-specific, with only a few that are actually compatible with each other. This is the realm of legacy operational technology (OT), where unlike in information technology (IT), we face an unlimited number of products, instead of only a handful pre-set platforms. The products are so specific that if the vendors don't provide enough documentation, system users will have a difficult time troubleshooting or monitoring for abnormal behaviors.

Those systems were operating fine before the introduction of the Internet. However, to keep up with the pace of growing technology, required efficiency, having to have to share data across different fields, and being able to give remote access to the corporate network, the remote fields and enterprise business systems integrating the Internet was inevitable. This integration caused an immense amount of vulnerability in these systems. Meanwhile, the newly developed modern security solutions were not compatible with these legacy ICS systems
.

The legacy industrial protocols have been designed without accounting for cyber risks or security mechanisms. Changing the existing protocols in these systems would cause compatibility issues with the underlying infrastructure, and most of the time requires re-designing the whole security architecture. Besides, providing security mechanisms for each protocol requires frequent changes and reinvestment. In addition, the ICSs are designed with a life span of decades, operating at full capacity, with little downtime and high reliability.

Therefore, it is more efficient to resort to a more flexible and better suited approach such as an intelligent intrusion detection technique to secure these systems.

Artificial intelligence (AI) would be the most cost efficient and yet effective tool to spot cyber attacks and malicious activities. AI models are wildly trained and used to detect malware, recognize abnormal patterns, and warn us of possible attacks before they enter our system. Using AI to secure the IIoT's network would remove the compatibility issues and introduces intelligent and adaptive way of detecting threats.

Since, AI runs in the system independently on the surface, it does not require any hardware compatibility. An AI-based security system is able to learn and improve its performance over time using only data rather than platform's specifics or being explicitly programmed. AI's potential in network security has been proven through years of experience in the research community. In the commercial world (e.g., product recommendation, social media, creative art, etc.), AI has been massively utilized in practice and shown to be successful. At the same time, interestingly enough, they have been scarcely deployed in real-world IIoT systems.

Several reasons prevent AI-based network security mechanisms from being successfully implemented. These reasons are due to the fact that basic assumptions of AI models are violated in security applications when it comes to use them in practical settings. It is important to solve these issues, not only to help the research community to be a true representative of real world scenarios but also to pave the way of utilizing AI in practice for IIoT systems.

In this dissertation, we aim to address some of the challenges in utilizing AI for real-world IIoT systems. We have identified key problems with IIoT security. In the next subsection, they are briefly introduced, while a chapter is dedicated to each to go into more details and discuss our endeavors in meeting them.

1.1 Challenges

In this dissertation, the focus is on three prominent challenges in utilizing AI for the security of IIoT. In a nutshell, these challenges are the reasons that hinder the research community from focusing on real-world IIoT scenarios or the security professionals from deploying AI in real-world IIoT settings. More specifically, these challenges are the lack of real-world data to train the AI, the black-box nature of the AI, and the high computational load of the AI. In the following, brief descriptions of each are mentioned.

1. Scarce Data: meaning lack of data that is a true representative of real-world IIoT networks.
    - Due to privacy issues, industries do not share their network data to public. This causes data scarcity in the research domain of IIoT.
    - Data scarcity has imposed great challenges in utilizing AI in practice. When facing rare events, it is hard to collect real labeled data when a scarce class of samples barely happens.
    - Training the learning models with imbalanced data, where the number of samples in one class is greatly larger than the other one has been a critical issue in this area.
    - This issue causes the model to naively label everything as the majority class because it did not learn the minority class properly due to lack of examples.
    - It is crucial to build models that are fair to all classes of data, especially rare events since they are usually the most important ones to detect.
2. Black-Box Nature: meaning only the inputs and outputs are visible and the internal operations are not visible/explainable/interpretable to the user.

- Another major reason for distrusting AI models in critical infrastructures is their "black box" nature. Analysts present the output of the model but cannot explain the reasons behind those conclusions. Thus, industry managements have a hard time understanding or trusting such outputs.
- The damage caused by a simple bug in the AI model and not knowing why it happened would result in incorrect operations of the critical infrastructure could halt the essential operations and have catastrophic results.
- Further, the advances in adversarial machine learning, which imposes falsified input to the model to manipulate its outputs, have made the AI adaptation even more questionable.
- In these IIoT critical applications, it is imperative that the conclusions made by the security system be explained and validated.

3. High Computational Load:
   - Network security solutions usually require executing computation-intensive and more importantly latency-sensitive techniques. ICSs are large systems and centralized processing of their enormous network data can lead to unacceptable delays.
   - As AI models become more sophisticated, they demand more computation power from the devices in the system.
   - IIoT devices have restricted computational resources such as memory, bandwidth, and power. Hence, they are unable to keep up with the security requirements on their own. Especially in the case of IIoT, where outdoor and on-field low-cost devices are utilized, the resource constraints are tighter.

- Processing power limitations have already started slowing down the utilization of AI. It is important to address this issue especially since having enough computing power is the main enabler of better performing AIs.

The primary goal of this dissertation is to address these three primary challenges for a successful real-world deployment of AI-based network security for IIoT systems. Therefore, we discuss the endeavors taken toward this goal through three main avenues which consist of developing successful AI models that are well-trained, transparent, and distributed despite data scarcity, lack of trust, and high computational load. In this work, many misconceptions about AI when it comes to network security applications are discussed. We also study how the learning models fall short when it comes to being trained with imbalanced datasets. Besides, we highlight the importance of building trust in AI models utilized in these systems and a potential method to achieve it. To avoid large unacceptable delays, our proposed AI models will be handled on a hierarchical paradigm to not leave a burden on the system by being distributed through the system's vein.

1.2 Contributions

The contributions of this dissertation can be divided into four categories. The first is more of an extensive study to provide background and comprehensive insights into the current status of network security of IIoT using AI and machine learning (ML). The other three are the research issues that we have solved.

1. A complete insight on AI-based security for IIoT:

    - We described the most common IIoT protocols, along with their main communication network vulnerabilities when utilized in practice.

- An network risk assessment of IIoT is conducted to identify common malicious threats, analyze the level of the associated risks, and their severity.
- We provide an extensive study on why AI/ML must be integrated into the security mechanisms and the cases where current algorithms fall short of providing the required level of security.
- We investigate the applicability of AI/ML techniques to counter the specific IIoT susceptibilities. This is an essential step to prioritize the required mitigation.
- We have reviewed the available research papers in the field that have designed AI/ML based security systems for IIoTs to familiarize with the state of literature and figure out how we can compliment them.
- We discuss not only where the current state of the art falls short, but also what steps we take to address these shortcomings, each of which will be mentioned in the following contributions.

2. To address the scarce data challenge:

    - With a complete picture in mind regarding the state of the art, we have studied an important characteristic of a realistic IIoT network dataset, imbalance, and different ways to overcome it.
    - We elaborate on the testbed developed at Washington University in St. Louis, to perform real-world IIoT operations, carrying out attacks (that have not been implemented either for the under-study or any other IIoT system by other researchers) against the system, and applying AI/ML based security solutions to tackle the intrusion problem, studying the most important features in identifying the attack traffic from normal.

- Details of our testbed, along with the collected dataset, their beneficial features helping to detect the attacks, how they resemble a real-world IIoT scenario, and how it sets them apart from the other publicly available datasets are thoroughly discussed.

- Presenting our own case study, the metrics that can fairly judge the performance have been compared to measure their effectiveness. We have also tested a technique that can be beneficial in overcoming the problem of imbalanced in network security applications.

- Our IIoT network security dataset has been released to support the research community for a more realistic and diverse data collection in different extensive sub-fields of AI for IIoT that there is an increasing interest in, such as explainable AI, distributed AI, etc.

3. To address the black-box nature:

    - We propose a universal, accurate, and well-performing XAI model called TRUST (Transparency Relying Upon Statistical Theory) XAI that requires no compromise in the selection or performance of the primary AI, while at the same time providing transparency in the primary model's results.

    - TRUST XAI is developed based on the statistical behavior of the underlying AI's outputs. The proposed technique is a surrogate model that runs in parallel with the primary AI.

    - Another desirable contribution of TRUST is that it is model agnostic and can work with any AI model and does not put any restrictions on the types of models it can work with.

- We show that TRUST XAI is very fast at reasoning the primary AI model's behavior on numerical data, making it one of the first XAI models suitable for real-time numerical applications.
- While maintaining an accepted level of performance, we present different means of trade-off between the speed and performance for reasoning behind the AI's decisions.
- We provide performance comparison not only against another popular XAI model, but also using three completely different datasets.
- The proposed XAI, TRUST, can be efficiently implemented in critical applications that deal with numerical data, such as IIoT networks, where low latency and high accuracy are a must.
- Presenting our own case study, we introduce a unique perspective on the assessment of AI-based trustworthy systems for IIoT and Industry 4.0.

4. To address the high computational load:

   - We propose a universal, accurate, and well-performing distributed AI (DAI) model called ADDAI (Anomaly Detection using Distributed AI) that does not compromise the learning performance. At the same time, it provides a fair distribution in the computational loads.
   - We remove the need of sending all the data to a central unit and making it the bottleneck of the system due to the high computational loads of AI.
   - ADDAI helps to detect anomalies close to the source of the data (the sensors) through a light-weight learner on the local processing unit. This speeds up the anomaly detection process since we do not need to wait for a central processor to alert us in case of a malicious intend.

- If the results on the local unit are not reliable or more investigation on the results are required, ADDAI takes advantage of those local decisions when it comes to a more sophisticated classification in the upper layer hierarchy.
- The proposed framework can be utilized as a low overhead intrusion detection system (IDS) for IIoT systems.
- The model is a scale-free framework that is capable of scaling up in hierarchy and out in as many as local unit as we want.
- Presenting our own case study, we introduce a unique perspective on the assessment of AI-based distributed systems for IIoT and Industry 4.0.

1.3 Dissertation Structure

All the research work in this dissertation have publish in either IEEE journals or peer-reviewed conferences. We have dedicated each chapter to two or one research paper. The structure of this dissertation is as follows:

- In **Chapter 2**, we mix the content of the two below papers. We present a background on using learning models for the security of IIoT, why it is necessary to use modern and intelligent solutions, their shortcomings, and what specific characteristics set them apart from other IT systems.

  [107]: Maede Zolanvari, Marcio Teixeira, Lav Gupta, Khaled M. Khan, Raj Jain, "Machine Learning Based Network Vulnerability Analysis of Industrial Internet of Things," in IEEE Internet of Things Journal, April 2019.

and

> [108]: Maede Zolanvari, Marcio Teixeira, Raj Jain, “Effect of Imbalanced Datasets on Security of Industrial IoT Using Machine Learning,” in Proceedings of the IEEE ISI (Intelligence and Security Informatics), November 2018.

- In **Chapter 3**, we again use some parts of these two papers. We provide details about our endeavors on addressing the challenge of lack of datasets that are real-world representative. We elaborate on details of our developed testbed, conducted attacks, gathered data, and built datasets. We present extensive evaluations on performance of different learning models and the effect of different imbalanced ratios in the dataset when training the models.

> [107]: Maede Zolanvari, Marcio Teixeira, Lav Gupta, Khaled M. Khan, Raj Jain, “Machine Learning Based Network Vulnerability Analysis of Industrial Internet of Things,” in IEEE Internet of Things Journal, April 2019.

and

> [108]: Maede Zolanvari, Marcio Teixeira, Raj Jain, “Effect of Imbalanced Datasets on Security of Industrial IoT Using Machine Learning,” in Proceedings of the IEEE ISI (Intelligence and Security Informatics), November 2018.

- In **Chapter 4**, presenting the paper below, we address the problem of black-box nature of the learning models. We discuss the details of our TRUST XAI model and empirically prove its superiority over another popular XAI model in performance and speed testing on three different cyber-security datasets.

  [109]: Maede Zolanvari, Zebo Yang, Khaled M. Khan, Raj Jain, and Nader Meskin, "TRUST XAI: A Novel Model for Explainable AI with An Example Using IIoT Security," in IEEE Internet of Things Journal , September 2021.

- In **Chapter 5**, presenting the paper below, we address the problem of high computational load of the learning models. Our developed DAI model, called ADDAI, which is a distributed anomaly detection model, is described. We present its success in achieving high performance and low communication overhead.

  [106]: Maede Zolanvari, Ali Ghubaish, and Raj Jain, "ADDAI: Anomaly Detection using Distributed AI," in Proceedings of IEEE ICNSC (International Conference on Networking, Sensing and Control), October 2021.

- In **Chapter 6**, we summarize the important points and conclude the dissertation.

Chapter 2

Security of IIoT Using AI

It is critical to secure the IIoT devices and the system as a whole because of potentially devastating consequences in case of an attack. AI, ML, and big data analytic are the two powerful leverages for analyzing and securing the IoT technology. By extension, these techniques can help improve the security of the IIoT systems as well. In this chapter, we first present the common IIoT protocols and their associated vulnerabilities. Then, we run a cyber-vulnerability assessment and discuss the utilization of learning models in countering these susceptibilities.

Following that, a literature review of the available security solutions using AI/ML models is presented. At the end, we conclude this chapter by summarizing the key points. This has helped use figure out which security control has been overlooked by the research community, so we would make sure to include it in our testbed and dataset.

Since IIoT security has its own specific traits, we need evaluation metrics that are able to describe the performance in a realistic way. We introduce different metrics that we will be using through out this dissertation and their equations. We, later in the following chapters, utilize these metrics to gauge the performance of the developed learning models.

2.1 Introduction and Motivation

As mentioned previously, the primary concept of the IIoT is to take advantage of IoT technology in the ICSs. ICSs are an integral part of critical infrastructures and have been utilized for a long time to supervise industrial machines and processes. They perform real-time monitoring and interaction with the devices, real-time collection and analysis of the data, and also gathering logs of all the events that happen in the industrial systems. Utilizing IoT technology in these systems enhances the network intelligence and security in the optimization and automation of industrial processes.

The supervisory control and data acquisition (SCADA) system is the largest subset of an ICS. It provides a graphical user interface (GUI) through its human-machine interface (HMI). The HMI makes it easy for the operators to observe the status of the system, interact with the IIoT devices, and receive alarms indicating abnormal behaviors. A general scheme of SCADA systems is shown in Figure 2.1. [2]

As shown in this figure, these systems consist of four different sub-systems; I/O network, supervisory control, control network, and corporate network. I/O network consists of the deployed IIoT devices (including sensors and actuators) in the industrial process. Supervisory control is the main sub-system responsible for securing, controlling, and monitoring the IIoT devices. The control network includes programmable logic controllers (PLCs) that directly sense and manage the physical processes. Since the sensors and actuators cannot

[2]A SCADA system can be a suitable representative of an IIoT system. By being a subset of ICS (therefore the IIoT), it includes all the essentials that an IIoT system has. As a result, from now on, in this dissertation, we may use the terms SCADA, ICS, IIoT interchangeably. By using any of these terms, we mean an IoT integrated SCADA system.

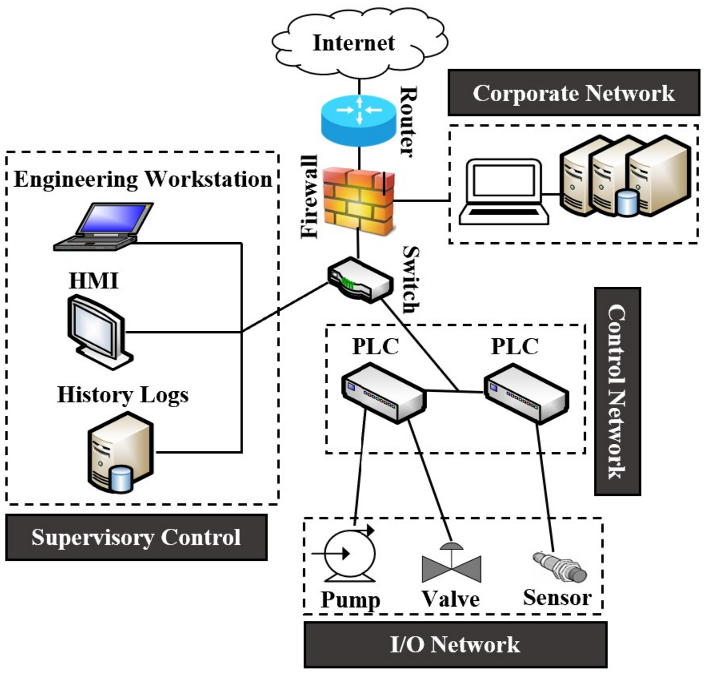


Figure 2.1: SCADA architecture

communicate directly, PLCs are used to collect the sensed data and send commands to the actuators. Finally, the corporate network consists of servers, computers, and other users connected to the network for other general services such as file transfer, website hosting, mail servers, resource planning, etc.

ICSs are mostly mission-critical systems with high-availability requirements. Their continuous operations lead to producing a huge amount of data that can be managed through big data analytic. In the past, these systems were standalone and isolated from the world, making them insusceptible to external malicious attacks. Recently, increased connectivity of ICS with corporate networks and utilization of Internet communications to transmit the information more conveniently have rendered these systems vulnerable to malicious attacks. Due to the sensitive nature of many industrial applications, security has become the primary concern in SCADA systems. More specifically, lack of security considerations in their communication protocols directly compromises the availability, safety, and reliability of these systems. Our work in this area shows that AI/ML based solutions can introduce new countermeasures to secure these systems.

2.2 IIoT Communication Protocols

There are several IIoT data transmission protocols used in the IIoT systems. However, most of these protocols have been designed without accounting for cyber risks or security mechanisms to counter them. The legacy of SCADA started with Modbus communication protocol which is still the most widely employed protocol in these systems. Recently, there has been a trend in moving towards newer protocols such as Building Automation and Control Network (BACnet), Distributed Network Protocol version 3 (DNP3) and Message Queuing Telemetry Transport (MQTT). All four are open protocols. Modbus was developed as a SCADA-vendor specific protocol in 1979 [40]. BACnet and DNP3 are standard protocols that were published in 1995 and 1993, respectively [17]. MQTT was developed in 1999 [62].

In this section, we study these four popular IIoT communication protocols along with their main security vulnerabilities. Since the main focus of this dissertation is on the network susceptibilities of the IIoT, this section has been included to show where each communication protocol is most vulnerable.

2.2.1 Modbus

Modbus is one of the earliest and the most commonly used protocol in the SCADA systems. The communication is serial and based on master-slave configuration. Master (e.g., HMI) is the device requesting the information and slave (e.g., PLC) is the one supplying the information. The master can also write data on the slave's memory registers. In a standard Modbus communication network, there is one master that can have up to 247 slaves, each with a unique identification code (ID). There are four tables stored in the slave device, two for storing digital data and two for numerical analog data.

Modbus does not provide confidentiality, authentication or integrity. Because all Modbus traffic is communicated in clear text (no encryption is provided), it lacks confidentiality, and the content of the packets can be easily seen using a sniffer tool. Modbus does not provide any public/private key management, which leads to lack of authentication as well. Also, there is no sufficient security check mechanism on the traffic, which makes it easy for the attacker to compromise the integrity of the data. Moreover, a flooding attack can interrupt the operation of the system and limit its availability.

2.2.2 BACnet

BACnet was primarily designed for building automation and control systems. This standard is currently under ASHRAE Standing Standard Project Committee (SSPC) 135. BACnet, like most of the other industrial communication protocols, was not designed with security considerations.

This standard provides several communication options, such as Ethernet, token-passing, master-slave, or point-to-point connections [10]. In SCADA, it is more common to use the master-slave mode of BACnet.

Due to lack of proper mechanisms for data confidentiality, reconnaissance attack is feasible. This protocol does not provide authentication procedures either. A few cryptography mechanisms, e.g., DES (Data Encryption Standard) and AES (Advanced Encryption Standard), have been included in the new versions of the BACnet standard. However, they are almost never utilized in industrial systems to maintain compatibility with the existing devices. Even in new green-field installations and in general, implementing encryption adds communication delays and large overheads in the system. The scan cycle times of PLC and HMI are usually on the order of milliseconds, and it is not yet feasible to accommodate data encryption in the system since encryption and decryption require a longer time process. In addition, denial of service (DoS) attacks can be conducted to target the system availability and halt the service [36].

2.2.3 DNP3

DNP3 is another standard network protocol used in SCADA systems. It was originally designed to be very reliable, but it does not provide sufficient security mechanisms. As a result, most of the DNP3 devices lack authentication, encryption, and access control. DNP3 covers the four OSI layers: network layers, application layer, data link layer, and physical layer. The communication can happen in point-to-point mode but mostly happens in master-slave configurations, and it can include multiple slaves and multiple masters.

No message authentication is deployed in this protocol, and hence data integrity is at risk. Eavesdropping and spoofing are easy as there is no encryption and the data is sent in clear text. DoS attacks can also easily impact the system's operation [100]. The latest version of this protocol that was published by IEEE in 2012, provides secure authentication, using IEC 60870-5 standard [38]. Through this standard, which has been developed for control systems, the authentication is provided using digital signatures. However, utilizing public key infrastructure (PKI) in IIoT devices is not feasible yet. The complexity that PKI adds cannot be handled by the simple IoT devices. Further, exchanging the keys, updating the keys, issuing or revoking certificates and other complexities that come with using PKI will add a huge delay in the system's performance.

2.2.4 MQTT

MQTT is an open standard under OASIS and is based on publish/subscribe configuration. This network protocol has been very popular in the IoT domain because MQTT messages are very simple and lightweight. Recently, MQTT has been increasingly adopted in the ICSs due to its suitability for remote sensing and control.

MQTT's topology consists of clients and brokers. At any particular time, each client can be either a publisher or a subscriber based on whether they are requesting or supplying data. A broker is an intermediate device between the publishers and the subscribers to filter out the published data and send them to their subscribers. Each broker can handle thousands of clients, which helps the system with scalability.

No encryption method has been implemented in MQTT, but TLS/SSL (Transport Layer Security/ Secure Sockets Layer) can be applied on the underlying TCP/IP (Transmission Control Protocol/ Internet Protocol) to provide an encrypted pipeline for the MQTT messages. However, since this requires a high level of complexity on the clients, it is not practical to use in the IIoT devices. Another main security drawback of the MQTT that originates from its topology is that, if an intruder steals the identity of a client, it will have access to all other clients' data, and not only that specific victimized client. On the other hand, the broker can be designed to ask each client for their username and password to allow them to join the network. However, these credentials will be transmitted in clear text, if no form of encryption is utilized. For data integrity, MQTT can provide MAC (message authentication code) techniques such as HMAC (hash-based MAC) to ensure the received data hasn't been tampered. HMAC is a lightweight cryptographic hash function. However, all clients who are aware of the secret key can sign or verify the data with the hash [35].

2.3 Prevalent Network Vulnerabilities and Cyber Threats

In this section, the nine most prevalent attacks in IIoT systems and the associated risks are studied. It should be noted that due to their fundamentally different nature, the prevalent vulnerabilities and security priorities in ICSs are different from the ones in traditional IT systems. Since there has been an extensive discussion on these differences, and this matter is beyond the scope of this dissertation, we refer the readers to the references [85] and [105].

The CIA triad (Confidentiality, Integrity, and Availability) are the security traits that must be preserved in any system to keep the data safe. Further, the AAA security controls (Authentication, Authorization, and Accountability) are the security tools to protect the CIA traits in the system. In this section, we study the most prevalent attacks targeting each of these security elements. Accountability has been left out since it is generally an administrative aspect.

Our extensive study of relevant works (such as [85], [76], [24], [21], and [60]) reveals that these are the most common threats in the SCADA systems. However, unlike the existing works, in this chapter, we

- provide a comprehensive set of prevalent attacks,
- define each attack separately regarding which security aspect they compromise,
- explain how they impact the IIoT performance,
- run a risk assessment based on the damage severity and the likelihood of happening in these specific systems, and
- study the effectiveness of AI/ML based security solution to encounter each class of attack.

2.3.1 Prevalent Attacks

The attacks are divided into five classes, based on which security aspects (integrity, availability, confidentiality, authentication, and authorization) are compromised. However, it is nearly impossible to define a solitary classification because the classes in which these attacks fall are not mutually exclusive. Often, compromising one aspect leads to compromising others as well.

2.3.1.1 Integrity

2.3.1.1.1 Buffer Overflow

In buffer overflow attacks, the intruder tries to write large data (more than the designated size) in the buffer, causing the extra bits to overflow and overwrite other buffers and alter their values. This attack is usually caused due to poor input type or size validation mechanisms and makes the system unreliable or even crash.

Buffer overflow attacks are highly prevalent in SCADA systems due to two main reasons. First, the majority of the operating systems in ICSs are written in programming languages such as C, which lacks type safety mechanisms. Further, SCADA devices operate continuously. The operating systems that have not been rebooted for years are more vulnerable due to accumulated memory fragmentation.

The buffer overflow problem in SCADA systems can affect both the supervisory control and field devices such as sensors. PLC's instructions to the output elements (e.g., turning on or off the water pumps) and sensed data (e.g., water level) could be manipulated through this attack [105].

2.3.1.1.2 Code Injection

In a code injection attack, the intruder tries to execute malicious commands or inject malicious data into the system. For instance, in a SQL injection attack, SQL queries are sent to control or compromise the database server. This attack exploits the system vulnerability due to the lack of user-supplied input data validation techniques.

This attack enables the intruder to access sensitive information such as usernames and passwords, and also alter the data (e.g., allowing access to an unauthorized user, deleting the data, etc.). A command injection attack can manipulate the control commands in the system and disrupt the normal operation.

Since the primary function of the SCADA systems is collecting and storing information, this attack may have a serious impact on the system. More specifically, if the system is controlled remotely through a web interface, this attack is able to compromise the data and the authentication procedures.

2.3.1.1.3 Improper Input Validation

This vulnerability is associated with the lack of proper mechanisms to validate the user's input. This is a more general type of vulnerability, which could lead to other types of risks. The attacker may be able to enter wrong values that can make the system unstable. Moreover, since these systems are not checked regularly due to their deterministic nature, this attack might stay undetected for a long time [24].

2.3.1.2 Availability

2.3.1.2.1 Denial of Service (DoS)

An intruder carries out a DoS attack to flood the targeted computer (e.g., PLC and HMI). This attack disrupts the availability of the SCADA system by sending a large number of random packets to the target node at a high rate to make the target unresponsive and may even crash the whole system.

A DoS attack against a SCADA system is generally carried out by an intruder connected to the network using SYN or HTTP flooding against a host. SYN attacks are constant fake synchronize requests, and HTTP attacks are either GET or POST requests to keep the web server of the target busy and not be able to respond to the normal traffic. If the links in the network are congested, monitoring and controlling the ICS will be highly difficult, if not impossible.

Therefore, the main goal of the DoS attack is to hurt the system's availability, so that legitimate users are not able to access the resources.

2.3.1.3 Confidentiality

2.3.1.3.1 Reconnaissance

In a reconnaissance attack, the intruder engages with the SCADA network to gather information about the system, such as the connected devices, security policies, IP addresses, host information, etc. After identifying the elements of the network, the attacker maps the network architecture to identify the vulnerabilities in the system. Eventually, the attacker may use this information to run exploits against susceptible devices to interrupt the system's functionality.

Intruders may start this attack using sniffers. They eavesdrop and inspect the ongoing network traffic to gain information about the network elements and their status. Stealth scan in SCADA network can occur on the link between any of the two nodes of the network; for instance, the link between the I/O network and the PLC or the link between the HMI computer and the PLC. This attack is considered passive since the attackers are silent and do not inject any traffic that would expose them. Although this attack may not be considered severe, the network information is exposed to an unauthorized person, and it is very difficult to detect.

2.3.1.4 Authentication

2.3.1.4.1 Unauthenticated Access

This vulnerability is due to poor authentication mechanisms in SCADA systems. Since these systems run continuously and autonomously, personnel may not change their usernames and passwords regularly. They may even use default usernames and passwords for ease of remembering [76]. Brute force methods or logging the user's keystrokes can be used to obtain this information. Furthermore, phishing attacks have been conducted widely to collect the credentials of ICS operators [24]. If the attacker somehow figures out these credentials, he can misuse his access and conduct other types of attack.

Since under this category, we solely consider "accessing" the data, in which usually root access is not granted. We have classified this attack as low impact. Otherwise, they will be categorized in more severe attack types such as directory traversal.

2.3.1.4.2 Man-in-the-Middle

In the man-in-the-middle attack, the intruder eavesdrops on the communication links and tries to compromise the messages between two nodes while the nodes think they are still talking to each other directly. For instance, the intruder may send malicious commands to the actuators pretending to be the PLC or send false responses from the sensors to the PLC. Further, the intruder may discard or manipulate messages. This type of attack will have a valid syntax code; hence, rule-based intrusion detection system (IDS) will not be able to identify it from the message format [93]. This type of attack can be mostly prevented through encryption techniques.

2.3.1.5 Authorization

2.3.1.5.1 Directory Traversal

In this attack, the intruder tries to access the restricted directories or files that are supposed to be root access only. This vulnerability is due to poor filtering or validation mechanisms for user-supplied inputs. Poor directory listing control is another cause of these attacks. In this type of attack, the intruder will be able to download sensitive files and information from the system.

This attack often also results in compromising other vulnerabilities in SCADA systems such as confidentiality, since the attacker might access private files in the system. Proper input validation methods can prevent this type of attack.

2.3.1.5.2 Backdoor

In a backdoor attack, the intruder tries to find a way around the authentication process to enter the system. Through the backdoor access, the attacker can log into the system, reach all the data and files on the system, and execute commands. Backdoor installation on the victim system may be done by an insider. Once installed, it is very difficult to detect this type of attack, and it is considered highly dangerous since it grants the intruder full access to the system.

In the case of ICSs, some of the vendors and manufacturers have backdoor accounts into their products for remote support and updates [12]. This vulnerability puts the system in danger, and in case of a successful attack, all the SCADA data will be exposed to the intruder.

2.3.2 Risk Assessment

As discussed in the previous subsection, different prevalent attacks have different severity and different rates of recurrence. We have built a risk assessment matrix for these vulnerabilities, which is shown in Figure 2.2. In this assessment, the impact and the likelihood of occurrence of the prevalent attacks are combined for integrated analysis. The matrix in this figure has been designed based on our study of the prevalent vulnerabilities and the severity of the risks associated with them. Vulnerabilities have been arranged in the matrix in the order of their likelihood and impact severity. It is important to mention that this order in the presented risk assessment matrix is based on our experience and judgment, and it might slightly differ from case to case or for different applications. However, it becomes a convenient tool for future studies.

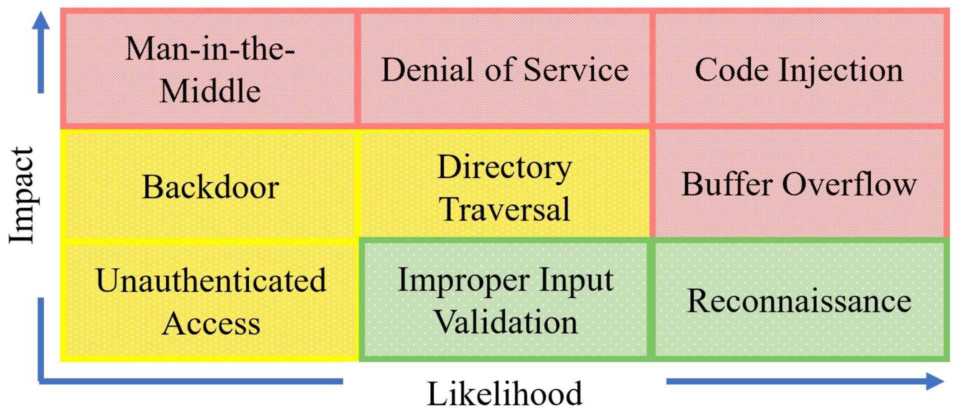


Figure 2.2: Risk assessment matrix of prevalent vulnerabilities in IIoT

Since we have picked the most common attacks, the likelihood is classified into three high levels of occurrence: occasional, likely, and certain. Similarly, the level of impact has been classified as mild, moderate, or critical. The overall risk ranking has been color coded. Threats that have severe negative impacts and are likely to occur frequently receive the highest rank, shown in the red color. Attacks with both low impact and low likelihood have the lowest rank, shown in the green color. And the yellow colored attacks fall between the two other classes. This risk assessment specific for the IIoT helps in identifying the threats that have the greatest overall risks and must be the top priority to address in these systems.

For instance, code injection is shown in red due to the catastrophic results of command manipulation in SCADA systems and their high probability of occurrence. DoS attacks that often occur would result in the termination of the system's operations. The reconnaissance attacks that harm the confidentiality may or may not lead to any negative consequences in the system's function.

2.4 AI/ML as a Versatile IIoT Security Tool

In this section, we go into more details regarding the reasons that why utilizing an intelligent learner is important and where it falls short. Later, we discuss the practicality of AI/ML models for each specific vulnerabilities of IIoT.

2.4.1 Why Intelligent Learning Models?

IDS as an effective mechanism to counter intrusions has been widely used to provide a secure platform. Rule-based, signature-based, flow-based, and traffic-based are just some examples of different ways that intrusion detection has been implemented. Regarding the IIoT system, as mentioned before, traditionally most of the connections and traffics in an ICS network were pre-defined. Hence, these types of IDS (ruled-based, signature-based, etc.) would detect abnormal activities very efficiently. For instance, when the intruder had to somehow manipulate the structure, like building new connections to the victims or sending a different type of traffic, ruled-based IDS would be successful in detecting the malicious attempt [80].

However, considering frequent upgrades in the networks, which results in regular changes in the topology, the legacy types of IDS will not work. Since these IDSs are designed based on defined topologies (e.g., allowed connections, allowed devices, etc.) any small changes in the system will raise a false alarm, unless the whole IDS would be re-designed after each change, which is an intensive task and might not work properly. On the other hand, the legacy IDSs cannot keep up with the attackers constantly evolving their methods. Furthermore, to counter new attacks that appear every day, or in scenarios where the attack is planned perceptively (e.g., the man-in-the-middle attack), intelligent IDSs are required. An anomaly-based IDS that employs ML algorithms can detect any out of the ordinary activity if the training procedures are handled correctly.

The intelligent IDS is based on the fact that AI algorithms can detect anomaly patterns that are difficult for a human to discover. Unlike rule-based IDSs, ML-based IDSs can successfully detect new types of attacks, different variants of a specific attack and unknown or zero-day attack. The zero-day exploit takes advantage of unknown vulnerabilities (i.e., the developers have no idea that they exist) to manipulate the processes or the system. These are all the reasons that ML should be applied in designing IDSs.

2.4.2 Where Intelligent Models Fall Short

Despite the confidence in the ability of these learning models to detect anomalies very effectively, there exist several challenges that arise when considering their applicability in IIoT. Without addressing these problems, the AI/ML algorithms are unable to function properly. In an IIoT environment, some of these challenges are manageable and some might not, due to their nature and associated security aspects.

The very first consideration is to choose proper features from the network traffic dataset. Sensor data in IIoT are usually obtained during an extended period from many sensors with different sampling frequencies, which results in high-dimensional datasets. Using raw data like this will add a large delay in training and detecting process. On the other hand, If the selected features do not vary during the attacks, even the best algorithm will not be able to detect an intrusion or an anomalous situation using that feature. It is, therefore, necessary to extract discriminating features to be able to use ML techniques. Applying power spectral density, Fourier analyses, linear feature extracting, and principal component analysis (PCA) are some examples that could be tested to reduce the dimensionality and find the most useful features.

On the other hand, due to the confidentiality and user privacy restrictions, industrial companies hardly release their protected network data on the intrusion attacks that might have occurred. Hence, training the algorithms on data collected from real networks of real IIoT's is almost impossible. Hence, most of the available research work in this area is done on commercial or public datasets that may not be specific to IIoT. That is another barrier in utilizing these AI/ML techniques directly in companies or industrial networks. Due to this reason, we built our IIoT testbed and took all consideration into account to make it as resembling as possible to a real industrial plant. More details are provided in the next chapter.

Furthermore, in any real world IIoT system, the number of intrusion attack samples is significantly low. Since the intruders do not wish to be exposed; they usually run their attacks randomly in short periods of time. This leads to a very low amount of attack data to train the learning model. This problem is known as an imbalanced training dataset. In other words, imbalanced dataset means the percentage of the attack traffic compared to the normal traffic in the whole dataset is very low.

Here, we briefly mentioned the challenges that are most critical when it comes to building a testbed and collecting a dataset for IIoT applications, to prepare a background for the next chapter. That being said, there are other challenges that must be considered when utilizing AI/ML techniques for intrusion detection, such as their black-box nature and their high computational load. We will cover these two challenges later in this dissertation, Chapters 4 and 5. Due to all these reasons, the suitability of ML under different circumstances must be considered.

In the following chapter, Chaper 3, we will talk about the imbalance challenge in more details. We will also discuss how we made our datasets to comply with this problem to be a better representative of a real-world IIoT system. For the rest of this chapter, we will focus on how AI/ML models can be beneficial for the IIoT's vulnerabilities that we discussed earlier.

2.4.3 Learning Models and IIoT's Vulnerabilities

IDS has been widely used as an effective security mechanism to counter intrusions. Misuse-based IDSs such as rule-based, signature-based, flow-based, and traffic-based methods are just some examples of conventional IDSs. Since traditionally, most of the connections and traffic in the SCADA networks were pre-defined; these types of IDS were successful in detecting abnormal activities. For instance, when the intruder builds new connections to the victim or sends a different type of traffic, there will be unusual data flows in the network [80].

However, considering frequent upgrades in the networks, resulting in regular changes in the topology, the legacy IDSs do not perform properly. Also, to counter new types of attacks that appear every day, or in scenarios where the attack is planned intelligently (e.g., the man-in-the-middle attack), smart IDSs are required.

IDSs are, in general, helpful whenever the intruder affects the network data flow. This is true even for AI/ML-based IDSs. If the intruder does not interact with any of the network elements, it is very difficult even to become aware of the intrusion. However, to launch attacks or compromise the network activities, the intruder has to disrupt the network somehow. The ability of ML algorithms in detecting small anomalies distinguishes them from any other type of IDSs.

AI/ML algorithms can detect anomaly patterns that are difficult for humans to discover. To provide a secure network, the AI/ML-based IDS can be designed with a moving target. This ability of learning models to learn and evolve is valuable because the attacks are constantly evolving, and new vulnerabilities are discovered every day. This is another reason why signature-based IDSs are becoming obsolete, and anomaly-based IDSs using AI/ML are the new trend. We now discuss the suitability of AI/ML-based IDS for each of the security elements.

2.4.3.0.1 Integrity

Learning models can be very helpful as a detection tool against data integrity threats. By training an AI/ML-based IDS with legitimate traffic data, the IDS will learn the normal data that flow in the system. For instance, in the case of command injection, the model will detect the malicious queries that are out of the ordinary in the system. This specialized IDS is able to recognize the source that is compromising the integrity of the data to block him from the system to maintain the trustworthiness of the data. Hence, by learning the common behavior of the system, the ML-based IDS can be very useful against the attacks targeting this security element.

2.4.3.0.2 Availability

AI/ML can be very useful in detecting the DoS attacks. A proper algorithm can detect specific characteristics of the attacks targeting the availability, for instance, detecting the sources with unfamiliar or broadcast addresses, the ones that are showing abnormal behaviors, nodes that are sending an unreasonable amount of traffic, or when the normal operational traffic stops because the HMI or the PLC are flooded and unavailable.

Even though a simple network analyzer can detect the DoS attacks, it still requires a human operator to analyze the network logs. On the other hand, the ML-based IDS will not only provide proper automation but is also not prone to human error. Moreover, it has been shown to be effective in detecting this kind of out-of-ordinary behaviors.

2.4.3.0.3 Confidentiality

In this type of attack, if the intruder merely eavesdrops on the network traffic (i.e., does not send any traffic nor build a connection with the devices in the network), it is very hard to detect using learning algorithms. As mentioned before, when the attacker's behavior does not change the network flow, it is very hard to detect the attack with any technique, including AI/ML. However, as soon as the intruder engages with the network, a machine learning powered anomaly-based IDS will be able to recognize the abnormal behavior of the attacker trying to snoop or asking for unusual information from other nodes in the network. However, after engaging with the network, the malicious activities go beyond a simple eavesdrop attack and are classified under other attack categories.

2.4.3.0.4 Authentication

As mentioned before, authentication is a security control technique. Attacks targeting this security element need to find a way around to bypass this step. To counter these threats, it will be more efficient to use prevention techniques rather than detection methods. For instance, encryption, strong passwords or key management techniques can be utilized to prevent unauthenticated access. Even though these techniques have their weaknesses, they improve the system's robustness against unauthenticated access.

2.4.3.0.5 Authorization

Activities that do not match with the normal traffic pattern even from verified users can be identified using learning techniques. Some examples include executing abnormal commands, manipulating the sensors and actuators, or sending random traffic on the network. If the intruder runs zero-day attacks or occasionally accesses the system, he might stay undetected for a while, but he will eventually be exposed by an AI/ML-based IDS. However, the sensitivity of the learning technique must be high. The IDS learns the normal conditions of the system and will reveal abusive commands, unauthorized users, or intruders.

The IDS would raise the alarm each time it detects an abnormal behavior from a user in the network that must be verified by the operator. Raising the sensitivity of the utilized AI/ML to detect these attackers will increase the number of false positives (normal traffic classified as attack traffic). Nevertheless, in security matters, it is better to be overcautious keeping the IIoT network safe. A false negative (undetected attack) could result in a higher cost than a false positive in the critical infrastructures.

2.5 Existing AI/ML Based IDSs

As we discussed in the previous section, since ICSs are different from regular IT systems, their communication type and even prevalent cyber vulnerabilities differ from a regular IT network. Consequently, it is important to consider these differences and design specific IDSs for SCADA systems.

In this section, we review available AI/ML-based IDS approaches solving different security vulnerability issues of IIoT systems. Some of the presented research works focus on various security aspects. In this case, the first time that we mention their work, a detailed description is provided; following that in other subsections, just brief mentions are made.

2.5.1 Integrity

Beaver et al. [14] employ six different types of ML algorithms, Naive Bayes (NB), random forests (RF), OneR, J48, NNge (Non-Nested Generalized Exemplars), SVM (support vector machines). Their dataset consists of labeled remote terminal unit (RTU) telemetry data from a gas pipeline system in Mississippi State University's Critical Infrastructure Protection Center. The attack traffic is generated from two types of code injection sets, command injection attacks, and data injection attacks. Seven different variants of data injection attacks were tried to change the pipeline pressure values, and four different variants of command injection attacks to manipulate the commands that control the gas pipeline.

Ullah and Mahmoud [92] suggest an IDS using a combination of J48 and Naive Bayes techniques. J48 is a type of DT (decision tree) technique. They have used the same dataset as the previously mentioned research work in [14]. The J48 classifier was first used as a supervised attribute filter. Then, the Naive Bayes classifier was used to develop the anomaly-based intrusion detection.

Alves et al. [8] employ the k-means technique, which is an unsupervised clustering algorithm. An open-source virtual PLC (OpenPLC platform) along with AES-256 encryption is used to simulate a SCADA system. They have conducted three different types of attacks against their system, code injection, DoS, and interception (eavesdrop).

He et al. [34] use CDBN (conditional deep belief network) to detect attacks in smart grids. They have simulated their system using IEEE 118-bus and 300-bus test systems. They also provide a comparison of their method with SVM and ANN (artificial neural network). They consider false data injection attack that is aimed at the integrity of data.

2.5.2 Availability

Potluri et al. [69] design a hybrid IDS using SVM, and DBN (deep belief networks) for industrial networked control systems. They have used the NSL-KDD dataset, which is an old dataset and is not specific to ICSs but consists of DoS and integrity attacks.

As mentioned in Subsection 2.5.1, paper [8] studies conducting DoS attacks as a part of their dataset to train their IDS.

2.5.3 Confidentiality

Keliris et al. [43] use SVM in their simulated testbed in MATLAB controlling Tennessee Eastman (TE) chemical process. They have conducted reconnaissance attack as vulnerability discovery technique, and further, tried to study the effect of command injection attack on the controller to manipulate the reactor pressure.

As mentioned in Subsection 2.5.1, paper [92] covers confidentiality problem through reconnaissance attack. Alves et al. [8] (mentioned in Subsection 2.5.1) also worked on eavesdropping attacks. Their unsupervised training methods were able to detect these attacks successfully.

2.5.4 Authentication

The research work [8] was mentioned previously. The authors also declare that due to the utilization of encryption, their system is resistant to man-in-the-middle attacks.

Eigner et al. [22] employ KNN (k-nearest neighbors) on a custom-built conveyor belt system. They use the normal behavior of the system to design an anomaly-based attack detection approach. They study the performance of the system with different k's and different distance measurement metrics. They have focused on man-in-the-middle attacks as the intrusion scenario.

2.5.5 Authorization

We could not find any work covering this area of intrusion detection for SCADA systems using ML techniques. Therefore, in this dissertation, we made sure we include unauthorized intrusion detection in our study. More specifically, we have conducted backdoor attacks on our SCADA IIoT testbed. The details are provided in Section 3.5.

2.5.6 No Specific Attack

Yasakethu and Jiang [101] study the advantages and disadvantages of four different types of IDSs, rule-based, ANN, HMM (hidden Markov model), SVM, OCSVM (one-class SVM). However, they do not provide any implementation nor any practical analysis of these techniques.

Zhang et al. [104] have used SVM to detect anomalies in network traffic. They have generated their dataset using simulations of the IEEE 118 bus network. Six features of traffic (magnitude and phase of the current, magnitude, and phase of voltage, real power, and reactive power) were extracted from the data. In that paper, the normal condition was defined as when no equipment is disconnected from the system, and there is no fault in operation. The fault condition is defined as a short circuit occurrence somewhere in the system. No cyber-attack was conducted against the system.

Skripcak and Tanuska [82] have designed a multi-agent architecture for SCADA systems to monitor the plant processes using passive-aggressive on-line ML algorithms. The focus of the paper is to provide the theory behind forecasting based on the current situation.

Siddavatam et al. [80] employ DT and RF techniques. Their system prototype has been built in their lab. They have extracted several features from the traffic for training such as TTL (time to live), byte count for response type, word count for query type, packet type, and the reference number for query type. To generate abnormal behavior in the system, changes in operation were conducted through a control node, but no attacks were developed in the system.

Maglaras et al. [50] use OCSVM as their proposed anomaly-based IDS. They declare that OCSVM is a good choice because the dataset is imbalanced. The authors have used only two features of traffic (data rate and packet size) of an electric grid. The trained model did not include any malicious attack data, and the trained dataset was captured during normal operation of a SCADA system.

Mantere et al. [52] have focused just on feature selections in designing an anomaly-based detection. They have chosen features such as flow directions, individual packet sizes, protocol, average packet rates, average data byte rates as the most determining features. The traffic was captured from two different locations within an industrial site. However, no attack data were considered in this paper.

2.5.7 Summary

Table 2.1 provides a summary of this section. In this table, the available ML-based anomaly detection approaches in SCADA are classified based on the type of their targeted vulnerabilities.

This table provides a concise overview of where the most focus of the research works available in the literature is.

Table 2.1: Available ML-based IDS for ICS vulnerabilities.

	SVM or OCSVM	**NB**	**DT or RF**	**DBN**	**ANN**	**KNN or K-means**
Integrity	[14] [34] [43] [69]	[14] [92]	[14] [92]	[34] [69]	[34]	[8]
Availability	[69]			[69]		[8]
Confidentiality	[43]	[92]	[92]			[8]
Authentication						[8] [22]
Authorization						
No Specific Attack	[101] [104] [50]		[80]		[101]	

As shown in this table, we could not find any ML-based research work in authorization aspects of SCADA security. Hence, we have made sure that we add cuber-attacks in our dataset that would target authorization specifically. We will discuss this further in the next section.

2.6 Performance Metrics

Traditionally, the performance of the learning models is measured by metrics which are derived from the confusion matrix. Table 2.2 shows the confusion matrix.

Table 2.2: Confusion matrix in IDS context

		Predicted Class	
		Classified as Normal	Classified as Attack
Actual Class	Normal Data	True Negative (TN)	False Positive (FP)
	Attack Data	False Negative (FN)	True Positive (TP)

The description of the matrix confusion parameters is as follows:

- True Negatives (TN): Represents the number of normal packets correctly classified as normal.
- True Positives (TP): Represents the number of abnormal packets (attacks) correctly classified as attacks.
- False Positive (FP): Represent the number of normal packets incorrectly classified as attacks.
- False Negative (FN): Represents the number of abnormal packets (attacks) incorrectly classified as normal packets.

Based on the confusion matrix, the metrics that are used in this dissertation to evaluate the performance of the AI/ML algorithms are as follows:

- Accuracy: Shows the percentage of the correctly predicted samples considering the total number of predictions. Accuracy's formula is shown in Eq. 2.1.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.1}$$

- False Alarm Rate (FAR): Represents the percentage of the regular traffic misclassified as attacks. FAR's formula is shown in Eq. 2.2.

$$\text{FAR} = \frac{FP}{FP + TN} \tag{2.2}$$

- Undetected Rate (UR): The fraction of the anomaly traffic (attack) misclassified as normal. UR's formula is shown in Eq. 2.3.

$$\text{UR} = \frac{FN}{FN + TP} \tag{2.3}$$

- Matthews Correlation Coefficient (MCC): Measures the quality of the classification. MCC shows the correlation agreement between the observed values and the predicted values. MCC's formula is shown in Eq. 2.4.

$$\text{MCC} = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{2.4}$$

- Sensitivity: Also known as the true positive rate. A sensitive algorithm helps rule out an attack situation with more confidence when the predicted data is labeled as "normal." While Sensitivity and the UR are complementary, each shows a different aspect of performance interpretation. If the focus is on minimizing FN, we would want to increase the Sensitivity of the model as much as possible (close to 100%), so that a smaller number of attacks stay undetected. Meanwhile, UR represents the fraction of these FN samples. Sensitivity's formula is shown in Eq. 2.5.

$$\text{Sensitivity} = \frac{TP}{TP + FN} \tag{2.5}$$

Accuracy (Eq. 2.1) is the most frequently used metric for assessing the performance of the binary classifiers. However, this metric is not sufficient for evaluation in scenarios with imbalanced classes (i.e., one class is dominant and has more training data compared to the other). In our case, which is an IDS scenario, the proportion of normal traffic to attack traffic is very high resembling a realistic dataset. This case is also valid where detecting rare anomalies is crucial like fraudulent bank transactions and identification of rare diseases. Therefore, in addition to the accuracy, we use other metrics to gauge the performance in a more meaningful way.

Just to present an example, let's assume we have trained a random forest model, and Table 2.3 shows the confusion matrix of how it performed on the data.

Table 2.3: Confusion matrix in IDS context

		Predicted Class	
		Classified as Normal	Classified as Attack
Actual Class	Normal Data	450561	30
	Attack Data	20	761

Based on these values, we get an accuracy equal to 99.98%, UR equal to 2.56%, and MCC equal to 96.81%, FAR equal to 0.006%, sensitivity equal to 97.43%.

2.7 Conclusions and Future Directions

The cyber-security of the IIoT devices is critical. There is still a huge gap in providing adequate security for these systems, which is why it is crucial to focus on the industrial aspect of the IoT technology. Model learning solutions and big data analytic have been widely used to ensure a secure platform in the IT systems. However, due to the fundamental differences and dissimilar priorities of ICS and the traditional IT systems, their prevalent cyber-risks are different. Thus, special attention is required to provide security for IIoT. Through our discussions and extensive studies, we have demonstrated the effectiveness of AI/ML for the security of these systems.

In this chapter, we first studied the four most common protocols used in SCADA IIoT along with their security susceptibilities. Afterward, we carried out a risk assessment of the most important and prevalent vulnerabilities of the SCADA IIoT systems, and how AI/ML-based solutions would be useful to combat them. Following that, a literature review on the existing anomaly detection approaches for IIoT systems using learning models was provided to present in which area there is still a need for providing more research study.

As a future direction, more through studies are required to find the other shortcomings in using learning models in practice. There can be intelligent models developed specially to detect the threats that were marked red in the risk assessment table. Since these attacks lead to catastrophic results, it is essential to pay diligent attention to them. Attacks that target authorization must be studied and experimented. Despite the fact that they occur infrequently in IIoT systems, the consequences are tragic that might paralyze the infrastructure.

Another future consideration can be regarding the unseen attacks. Based on our findings, the threats will never stop. They might change or evolve over time and we will always face new threats. It is important to develop quality assurance measures when using AI/ML models to be included along with the modern security controls. This way, we can make sure that learning models would evolve too as the threats evolve.

Chapter 3

Data Sparsity for AI

ML and AI algorithms have been shown to be suitable for securing platforms for IT systems. However, due to the fundamental differences between the IIoT and regular IT networks, a special performance review needs to be considered. The vulnerabilities and security requirements of IIoT systems demand different considerations.

As mentioned in the previous chapter, since industrial settings do not release their network data to the public or the research community, we, the researchers in this field, are forced to use other non-IIoT specific cyber datasets to train our algorithms and collected insights about these systems. Hence, most of the available research work in this area is done on commercial or public datasets. In this chapter, we will discuss how we have addressed this issue.

We saw a need for better representing datasets, so we have built a lab scale testbed the complies with the specifics of a real IIoT systems. The challenges and real-world considerations associated with this matter are studied in our experimental design. We have developed an IIoT testbed resembling a real industrial plant to show our proof of concept.

On the other hand, as discussed in Section 2.5.7, attacks that target the “authorization” aspect of security controls have not been studied previously by other research works. Thus, we made sure we to include this kind of threats in our developed dataset through running backdoor attacks against the testbed.

In this chapter, we run experiments to compare different learning models. We have also studied the challenge of imbalanced datasets with different imbalanced ratios. This is an important characteristic of IIoT systems that must be taken into account when developing datasets to be representatives of real-world scenarios.

3.1 Introduction and Motivation

Data scarcity has imposed great challenges in utilizing AI in practice. When facing rare events, it is hard to collect real labeled data when a scarce class of samples barely happens. Training the learning models with imbalanced data, where the number of samples in one class is greatly larger than the other one has been a critical issue in this area. This issue causes the model to naively label everything as the majority class because it did not learn the minority class properly due to the lack of examples. It is crucial to build models that are fair to all classes of data, especially rare events since they are usually the most important ones to detect.

In recent years, there has been a growing trend in the data annotation tools market. This market is expected to reach 2.57 billion dollars by 2027, expanding at a CAGR of 26.9% from 2020 to 2027, according to a new report by Grand View Research, Inc. [31]. This shows how the research community is aware of this shortcoming and the AI’s hunger for more data is not going to end any time soon.

Since intrusion is the primary security concern in IIoT, an intrusion detection system (IDS) is an integral part of these applications to provide a secure environment. Stuxnet worm, which was exposed in 2010 [96] and recently reappeared (late December 2017), and Triton malware against the ICSs [29] raised awareness of the necessity for special attention to the security of these critical infrastructures. Due to the fundamental differences between the ICSs and the regular IT systems, their common vulnerabilities and priorities are different [86]. Furthermore, ICSs have a specific type of traffic and data using particular IIoT communication protocols (e.g., Modbus, BACnet, DNP3). Due to all these reasons, proper diligence must be considered when it comes to designing a testbed and collecting data to build datasets for the IIoT systems.

AI/ML-based security solutions have been widely used in providing security for IT systems. However, the suitability of utilizng these techniques in their basic forms and the traditional way of evaluating them for IIoT applications is debatable. The main security concern in IIoT devices is to detect any penetration into the system. Intrusion detection comes with special features such as significant imbalanced datasets that sometimes the trained algorithms may not be able to detect the attack.

In this chapter, first, we discuss the critical characteristic of the network security datasets in real-world settings which is imbalanced dataset. Then, we present how we address the problem of data sparsity in this research field by building our testbed. The details of our testbed and different attack scenarios conducted against the testbed, such as denial of service (DoS), command injection, backdoor, and reconnaissance are provided. At the end, we evaluate different learning models and also different imbalanced ratios to gauge the performance under different circumstances. The metrics that can fairly judge the performance have been compared to measure their effectiveness.

3.2 Related Work

In this section, we review some of the related research works related to the network security of IIoT and their utilized datasets and we analyze their imbalanced ratios. To the best of our knowledge, imbalanced IIoT dataset problem with the significantly low number of minority samples has not been studied yet.

The intrusion detection problem in smart grids using several different ML techniques has been studied in [70]. Some countermeasures to overcome the problem of the imbalanced dataset have been examined. They have used ADFA-LD dataset that consists of 12.5% attack data. It is important to notice that this ratio is not realistic in the case of IIoT applications. Here we deal with less than 1% anomaly samples in our applications, which makes the results closer to real-world scenarios. Various sampling techniques to overcome the imbalanced dataset problem have been investigated in [79]. The utilized datasets are extracted from Github and Sourceforge projects with 15% imbalance ratio. This work is not cybersecurity nor IoT related, though it represents a practical case study on this imbalanced dataset problem.

An IDS using a combination of J48 and Naive Bayes techniques is designed in [92]. The dataset was built using gas pipeline system of the Distributed Analytics and Security Institute, Mississippi State University, Starkville, MS. Their dataset consisted of different types of attacks such as reconnaissance, code injection, response injection, command injection. The J48 classifier was first used as a supervised attribute filter. Then, the Naive Bayes classifier was used to develop the anomaly-based intrusion detection. The ratio of attack traffic in their study was about 21.87%, which is far higher than a real-world case.

Six different types of ML algorithms, Naive Bayes, Random Forests, OneR, J48, NNge (non-nested generalized exemplars), SVM (support vector machines) for IDS have been studied in [14]. J48 is a type of decision tree technique. Their dataset consists of labeled RTU telemetry data from a gas pipeline system in Mississippi State University's Critical Infrastructure Protection Center. The attack traffic is generated from two types of code injection set, command injection attacks, data injection attacks. Seven different variants of data injection attacks were tried to change the pipeline pressure values, and four different variants of command injection attacks to manipulate the commands that control the gas pipeline. They used precision and recall metrics to make sure to have a fair evaluation in spite of the imbalanced dataset with about 17% attack traffic.

K-means technique, which is an unsupervised clustering algorithm, for IDS has been employed in [8]. An open-source virtual PLC (OpenPLC platform) along with AES-256 encryption is used to simulate an ICS. They have conducted three different types of attacks against their system, code injection, DoS, and interception (eavesdrop). However, they have not provided any information on the percentage of attack data that was used for training.

One class SVM (OCSVM) as a proper anomaly-based IDS has been proposed in [50]. They declare that OCSVM is a good choice because the dataset is imbalanced. The authors just used two features of traffic (data rate and packet size) of an electric grid. The trained model did not include any malicious attack data, and the trained dataset was captured during normal operation of a SCADA system.

3.3 Imbalanced Datasets

Even though ML has proven its capability in intrusion detection, there are cases that it fails heavily. Severely imbalanced training datasets where the number of attack data is significantly lower than normal data (e.g., less than 1%) is a real-world challenge that is quite common in IIoT security.

ML techniques like other AI classifiers generally perform best on balanced datasets. The problem of imbalanced datasets, specifically those in severe cases (i.e., significantly low number of samples from one class compared to the other), is a critical issue in the training process. Examples of such cases include detecting rare anomalies like fraudulent bank transactions and identification of rare diseases. Intrusion detection, which is the main security concern of IIoT applications, is another case that suffers from the severely imbalanced dataset. Due to the large amount of sensed data from IIoT devices (i.e., a large amount of normal traffic) on the one hand; and random, rare attack traffic (i.e., a small amount of attack traffic) on the other hand, the IIoT's security suffers greatly from imbalance problem.

There have been countermeasures suggested for this problem, through changing the sampling method. Under-sampling, over-sampling, or a combination of both are some examples. However, each of these techniques comes with several drawbacks. In simple terms, under-sampling means including fewer instances from the majority class, and over-sampling means including more samples of the minority class. One problem with under-sampling is the possibility of losing useful information, while over-sampling might cause overfitting problems. These techniques can be very complex, and these details are out of the scope of this dissertation. Since, in a real IIoT system, any of these techniques might lead to an unrepresentative model, the resulting models may not be accurate to solve the intrusion detection problem in practice. Plus, they result in different outcomes compared to the models trained with the full dataset.

However, to study an oversampling technique, we have chosen a popular one called synthetic minority over-sampling technique (SMOTE). In the following subsection, we give an overview of this technique.

3.3.1 Over-sampling

It is well known that the most widely used approach to synthesizing new samples for the minority class is SMOTE [15]. In SMOTE, a sample is randomly chosen, then we select a group of k nearest neighbors that are similar to our chosen sample in the feature space. One of these neighbors is randomly selected and we draw a lines between them and make a new sample at the middle point of that line. The typical value of k is 5, but it can be tested as a hyper parameter of the model. The number of synthesized samples should be decided based on how many we need for the minority class so the dataset would be somewhat balanced. Figure 3.1 depicts how this technique works. In this figure, $k = 4$ and the red sample shows the synthetic sample that was produced.

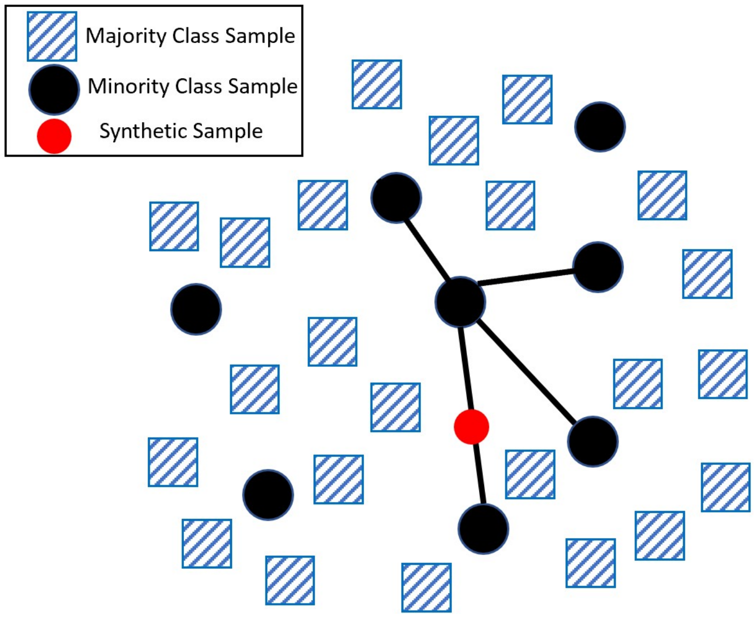


Figure 3.1: The SMOTE technique

In the following, we will discuss our developed testbed and how we ensured the imbalanced ratios to be adjusted in it so it would represent a real-world scenario. Later in this chapter, we present our investigation on using different learning models as IDS and studies on severely imbalanced datasets in IIoT systems and how AI/ML performs.

3.4 Our Testbed

In this section, we describe our testbed and the evaluation results of our ML-based IDSs to explore the problem of intrusion detection in the IIoT systems. To detect manipulated commands, system's transactions were logged and used to train the ML algorithms. We start this section by pointing out the extra steps we took to make the testbed even more inclusive of the features that an IIoT has. Afterward, we move onto the details of our testbed implementation, conducted cyber-attacks.

1. We have integrated not only digital sensors (i.e., the water level sensors), but also added analog sensors. Turbidity alarm and turbidity sensor have been embedded in the testbed to add analog input to the system. This helps us investigate overwriting analog registers (turbidity level of the water) in the PLC. A real-world IIoT system has a variety of sensors for both analog and digital data. By incorporating analog signals into our testbed, we have produced a more comprehensive scenarios.
2. We have investigated the IDS design using different AI and ML models for protecting different aspects of security controls by conducting different cyber-attacks against the testbed such as:
    - integrity by running command injection attacks
    - availability by running different kinds of DoS attacks
    - confidentiality by conducting reconnaissance attacks

- authorization by running backdoor attacks [3]

3. Fair metrics have been used to evaluate the performance of the trained model despite the fact that we deal with imbalanced datasets in these systems. These metrics represent the quality of the performance with better granularity.

4. Feature importance ranking study has been conducted to show which features are the most salient ones in distinguishing the attack traffic from the normal in an IIoT network data.

3.4.1 The Implementation

We have picked an IIoT system that supervises the water level and turbidity quantity of the water storage tank, shown in Figure 3.2. This system is a part of the water treatment and distribution process in industrial reservoirs. However, almost the same liquid treatment is utilized in any industrial field that deals with a liquid substance, such as oil and gas, pharmaceutical, petrochemical, pulp and paper, etc.

This testbed includes components like historical logs, HMI, PLC. There are three sensors and four actuators in this testbed. Two digital water level sensors and an analog turbidity sensor compose the inputs. A three-light turbidity alarm, a valve, and two water are the actuators that receive the commands from the PLC. Also, there are control buttons (On, Off, Light Indicator) for manual control of the system.

The main purpose of HMI in an ICS is to make it easy for the operators to observe the status of the system, interact with the IIoT devices, and receive alarms indicating abnormal

[3]Please bear in mind that, as mentioned in Section 2.4.3.0.4, we have not included attacks targeting authentication in our dataset. As discussed in that section, these threats are less suited to be detected by the learning models, and preventative measures are far more successful to ensure safety against them.

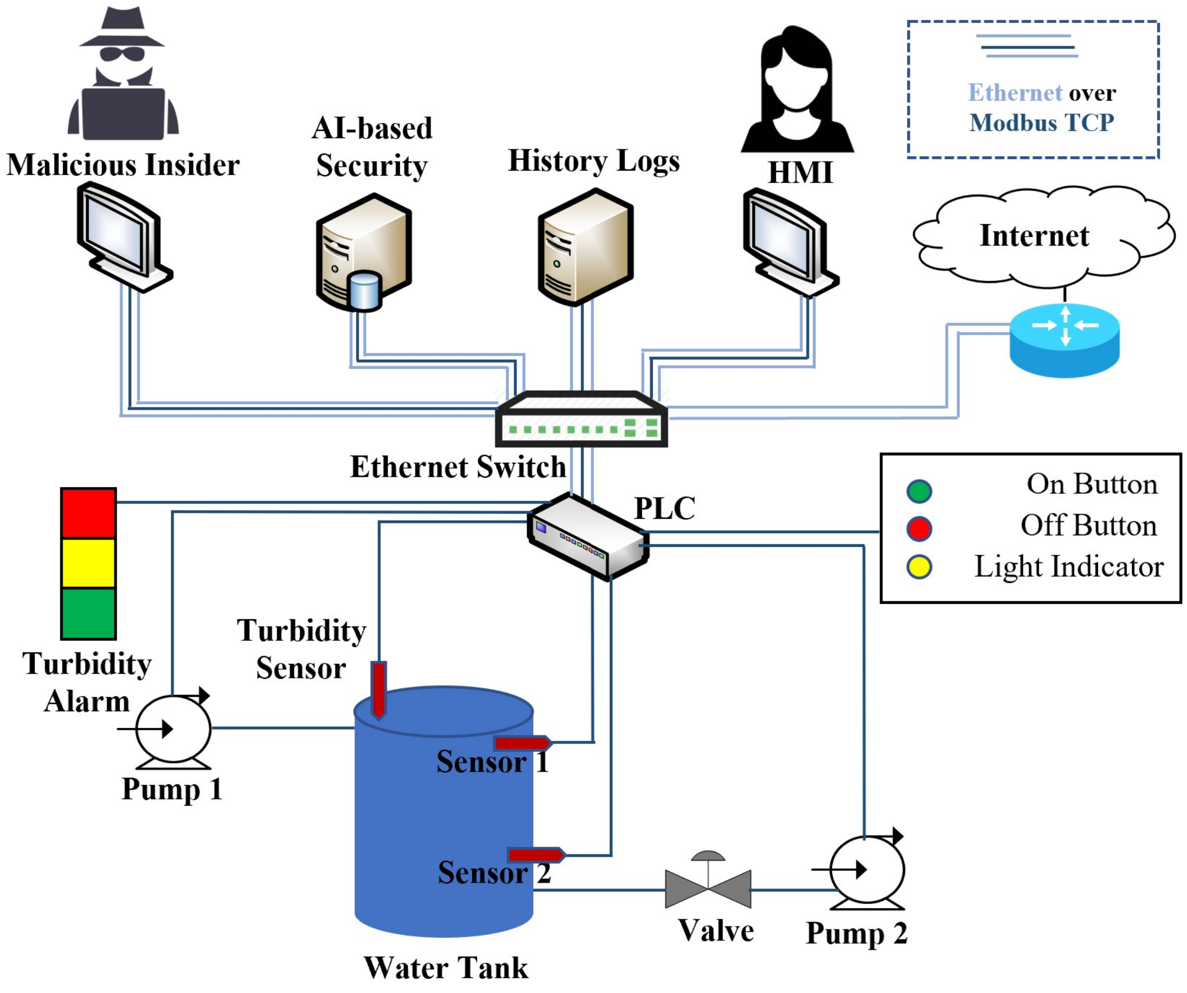


Figure 3.2: Scheme of our implemented testbed

behaviors. Moreover, since the sensors and relays cannot communicate directly, PLCs are used to collect the sensed data and send commands to the actuators.

The water storage tank has two level sensors: Sensor 1 and Sensor 2, which are used to monitor the water level in the tank. When the water reaches the maximum defined level in the system, Sensor 1 sends a signal to the PLC. The PLC turns off the water Pump 1 that is used to fill up the tank, opens the valve, and turns on the water Pump 2 draws water from the tank. When the water reaches the minimum defined level in the system, Sensor 2 sends a signal to the PLC. PLC closes the valve, turns off the Pump 2, and turns on the Pump 1 to fill up the tank. This process starts over again when the water level reaches the maximum level. Meanwhile, there is an analog turbidity sensor integrated into the system to measure the turbidity of the water. Based on two defined thresholds, PLC illuminates one of the red, yellow or green lights of the Alarm, in which green means the water has an acceptable level of turbidity, red means the turbidity is beyond the acceptable thresholds, and yellow falls between the two thresholds.

This IIoT testbed takes the data from sensors, and the status of the system from the PLC using the Modbus communication protocol and displays them to the operator through the HMI interface. Since Modbus is one of the most popular IIoT protocols, and it's widely used by large industries, we chose this protocol. The PLC model used in our testbed is Schneider Electric Programmable Logic Controller model M241CE40. The analog expansion module is TM3AM6 Modicon I/O Module. The logic of the PLC is programmed using the Ladder language [77], [23]. The turbidity sensor is SEN0189, and the water level sensors are Autonics CR18-8DP sensors. The deployed water pumps are GA-2328ZZ uxcell pumps.

3.5 Our Attack Scenarios

Since, to the best of our knowledge, no research paper has focused on machine learning-based IDS in SCADA systems for backdoor attacks, we conducted these attacks along with two other types of cyber-attacks. We have generated command injection attacks, reconnaissance, and DoS to have a larger variety of attack records in our dataset, and also, to test out other aspects of security measures. These attacks were carried out using the Kali Linux Penetration Testing Distribution [41]. All the data generated during the attack phase as well as the normal traffic was gathered and recorded by Argus [11] and Wireshark [97] network tools. A brief explanation of how these three attacks disrupt the normal operation is provided next.

3.5.1 Command Injection

In this attack, the target is the PLC. First, the attacker's PC connects to the network and is able to read all the PLC register values and logs them into a .txt file. After gaining access to the PLC register information, the attacker rewrites some of the PLC registers that are vital to the physical process. For example, while Pump 2 was supposed to draw water from the tank, we (as the white hat attacker) stopped it, started Pump 1, and the water flowed out from the tank. Another instance is when we turned on the wrong turbidity alarm light, in the way that, while the turbidity level was high, and the red light was supposed to be on, we (as the white hat attacker) turned off the red light and turned on the green light instead.

3.5.2 Denial of Service

Denial of Service (DoS) attacks target the availability of the PLC. By bombarding the PLC, the intruder would be able to make all the PLC's services unavailable. Since PLC is the center of the interconnection of different elements in the IIoT, this attack can result in a severe impact on the system's production. An example of this attack is SYN flood, in which an enormous number of initial connection request packets are sent to the PLC and HMI ports. This causes the target device to respond to the legitimate requests sluggishly or not at all.

3.5.3 Reconnaissance

Reconnaissance is the first stage of an intrusion plan. In this stage, by using network scanning tools, the intruder tries to figure out the target network's topology. The goal of this attack is to get a list of the devices employed in the target network as well as their specific vulnerabilities. The information that the attackers tries to gather about topology includes network addresses, identification of the Modbus server address, the Modbus slave IDs, and additional information such as vendor and firmware. By gathering this information, they can be used to conduct further attacks. Using the Nmap tool, packets are sent to the target at intervals varying from 1 to 3 seconds. The TCP connection is not fully established, so that the attacks are difficult to be detected by traditional ruled-based IDSs.

3.5.4 Backdoor

In this attack, our target is the HMI system, which gets infected with a backdoor virus. This virus works in the background and is hidden from the SCADA system's operator. The backdoor virus opens a port in HMI allowing a remote connection to be established with the attacker's PC. Thus, the attacker gains full access to the HMI computer, where the SCADA system is installed. Using the backdoor, the attacker can explore the HMI system and download any file, including the dataset with all the sensor and actuator values. In this attack scenario, we (as the white hat attacker) transferred about 1GB of files containing sensitive information from the HMI to the attacker's PC. We ran this attack several times to get the status of the system in different situations. Further, using this attack, we built new directories in the SCADA system and removed several files to disrupt the HMI operation.

3.6 Feature Selection

An important step in training the learning models is selecting and extracting features from the traffic. Here, in designing our IDS, we chose the features that their values change during the attack phases compared to the normal operation phases. If a selected feature does not vary during the attacks, then even the best algorithm will not be able to detect an intrusion or an anomalous situation using that feature.

In our study, we reviewed the potential features and chose 41 features that are common in network flows and also change during the attack phases. Table 3.1 shows the chosen features along with their description.

How each feature varies depends on the type of attack. For instance, during the normal condition, where there is no attack, the SrcPkts and DstPkts features mostly show a periodic behavior. On the other hand, during attacks, these features show random behavior.

Table 3.1: Selected traffic features to build our dataset

Features	**Type**	**Descriptions**
Mean flow (mean)	Float	The average duration of the active flows
Source Port (Sport)	Integer	Source port number
Destination Port (Dport)	Integer	Destination port number
Source Packets (Spkts)	Integer	Source/Destination packet count
Destination Packets (Dpkts)	Integer	Destination/Source packet count
Total Packets (Tpkts)	Integer	Total transaction packet count
Source Bytes (Sbytes)	Integer	Source/Destination bytes count
Destination Bytes (Dbytes)	Integer	Destination/Source bytes count
Total Bytes (TBytes)	Integer	Total transaction bytes count
Source Load (Sload)	Float	Source bits per second
Destination Load (Dload)	Float	Destination bits per second
Total Load (Tload)	Float	Total bits per second
Source Rate (Srate)	Float	Source packets per second
Destination Rate (Drate)	Float	Destination packets per second
Total Rate (Trate)	Float	Total packets per second
Source Loss (Sloss)	Float	Source packets retransmitted/dropped
Destination Loss (Dloss)	Float	Destination packets retransmitted/dropped

Total Loss (Tloss)	Float	Total packets retransmitted/dropped
Total Percent Loss (Ploss)	Float	Percent packets retransmitted/dropped
Source Jitter (ScrJitter)	Float	Source jitter in millisecond
Destination Jitter (DrcJitter)	Float	Destination jitter in millisecond
Source Interpacket (SIntPkt)	Float	Source interpacket arrival time in millisecond
Destination Interpacket (DIntPkt)	Float	Destination interpacket arrival time in millisecond
Protocol (Proto)	Char	transaction protocol
Duration(Dur)	Integer	record total duration
TCP RTT (TcpRtt)	Float	TCP connection setup round-trip time, the sum of 'synack' and 'ackdat'.
Idle Time (Idle)	Float	time since the last packet activity. This value is useful in real-time processing, and is the current time - last time.
Sum (sum)	Integer	total accumulated duration of aggregated records
Min (min)	Integer	minimum duration of aggregated records
Max (max)	Integer	maximum duration of aggregated records
Source Diff Serve Byte (sDSb)	Integer	Source different serve byte value
Source TTL (sTtl)	Float	Source $\rightarrow$ Destination TTL value
Destination TTL (dTtl)	Float	Destination $\rightarrow$ Source TTL value
Source App Byte (SAppBytes)	Integer	Source $\rightarrow$ Destination application bytes
Destination App Byte (DAppBytes)	Integer	Destination $\rightarrow$ Source application bytes
Total App Byte (TotAppByte)	Integer	total application bytes
SYN & Ack (SynAck)	Float	TCP connection setup time, the time between the SYN and the SYN_ACK packets

Run Time (RunTime)	Float	total active flow run time. This value is generated through aggregation, and is the sum of the records duration.
Source TOC (sTos)	Integer	source TOS byte value
Source Jitter (SrcJitAct)	Float	source idle jitter (mSec)
Destination Jitter (DstJitAct)	Float	destination active jitter (mSec)

Further, we have studied the importance of the features. They are ranked based on how salient they are in helping the algorithm distinguish the normal traffic from the attack traffic. In this technique, the values of each feature are permuted randomly one at a time, creating new datasets. The ML model is trained on these datasets, and the increase in classification error is measured for each. If the increase is high, then the feature is important, and conversely, if it is low, the feature is considered as not important. For each feature, the "model reliance" or importance coefficient is defined as the ratio of the model's error value after permutation to the standard error value when none of the variables are permuted. For more detailed information, we refer readers to [25] and [58].

As we report later in the results, random forest (RF) has shown the best classification performance, so we have picked this algorithm to calculate the importance. In Figure 3.3, the top five important features in our dataset along with their normalized (so the total of 41 feature importance values sum to 1) importance coefficient are shown. While these are the top five features, the threshold for the importance has shown that all the 41 features are relevant enough to be used for training.

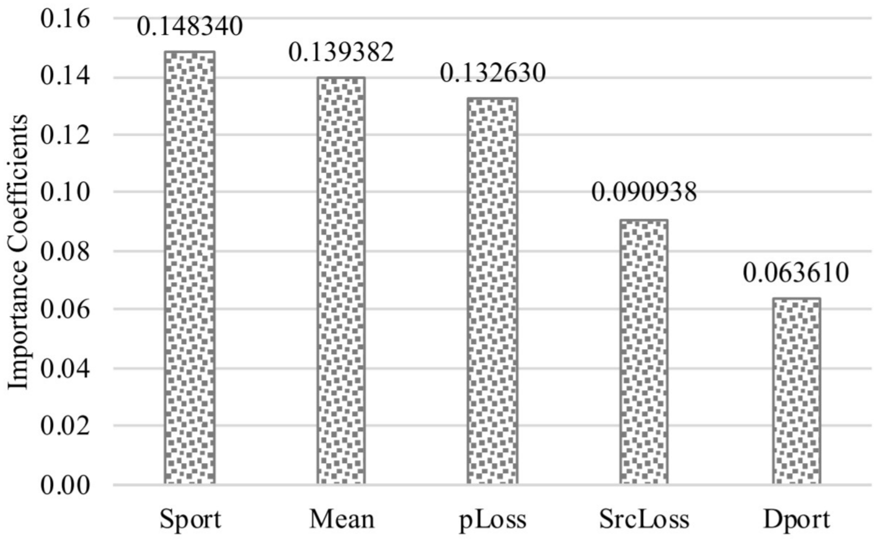


Figure 3.3: Top five important features

3.7 WUSTL-IIoT-2021 Dataset

The dataset collected from our testbed is called WUSTL-IIoT-2021, which we refer to as “WUSTL-IIoT" throughout this dissertation. We collected a 2.7 GB of data, for a total of about 53 hours. We have pre-processed and cleansed the dataset (removed the rows with missing values, corrupted values (i.e., invalid entries), and extreme outliers. The dataset that we have utilized throughout this dissertation is a smaller version of that which is a little over 400 MB.

Although, all the samples are labeled with the type of attack they belong to, to simplify, we have transformed the problem into a binary classification by labeling all the attack traffic as class 1 and normal traffic as class 0. Specifics of our dataset are in Table 3.2.

Table 3.2: Specifics of the developed dataset

Dataset	WUSTL-IIoT
Number of observations	1,194,464
Number of features	41
Number of attack samples	87,016
Number of normal samples	1,107,448

An important point that should be mentioned here is that we have deliberately built our dataset to be imbalanced. The percentage of attack traffic in the dataset is less than 8%. This assumption makes the system as similar as possible to the real-world industrial control systems. The statistics of the dataset are shown in Table 3.3, where the average data rate was 419 kbit/s, and the average packet size was measured as 76.75 bytes.

Table 3.3: Statistical information of the traffic types in our developed dataset

Type of the Traffic	**Percentage (%)**
Normal Traffic	92.72
Total Attack Traffic	7.28
Percentage of each attack	
Command Injection Traffic	0.31
DoS Traffic	89.98
Reconnaissance Traffic	9.46
Backdoor Traffic	0.25

Since DoS attacks are usually heavy in traffic and the number of samples, we deliberately devoted about 90% of the attacks to them. Other types of attacks happen less frequently and when they happen, they send only a few number of traffic data.

3.8 The Learning Models

In this chapter, the AI/ML-based IDS is designed just as a binary classification to decide whether a particular traffic sample is an attack or normal. The inputs to the IDS are the 41 chosen features, as mentioned in the previous section, and the output of the IDS is either 0 (normal traffic) or 1 (attack traffic). Also, in the rest of this dissertation, we use the ratio of 80% to 20% to randomly divide the dataset into training and testing sets, respectively.

We have used and tested seven different techniques for the IDS; SVM, KNN, Naive Bayes (NB), RF, Decision Tree (DT), Logistic Regression (LR), and Artificial Neural Network (ANN). We have used the Keras library [44] to build the ANN, and for the other algorithms, the scikit-learn library [78] was utilized to develop the learning models for the IDS. The models are trained and tested over the data collected in the testbed, and the results of their performance are compared.

3.9 Evaluation

We start this section by, first, training and applying several different learning models on our dataset as a way of exploratory data analysis (EDA) to see how different AI/ML based security models perform on our dataset. Afterward, we test out different imbalanced ratios to evaluate the practicality of the learning models in severe cases.

Please keep in mind that as shown in Table 3.3, we have built our dataset with regards of the attributes of a real IIoT system. One includes the dataset must be imbalanced, meaning the ratio of the number of attack samples to normal samples must be small. Therefore, the results of different performance metrics (discussed in Section 2.6) might not be as we intuitively expect them to be.

3.9.1 Comparing Different Learning Models

Here, we present the numerical results of different algorithms detecting the attacks described in Subsection 3.5. Figure 3.4 shows the accuracy results (Eq. 2.1). While RF shows the best performance and NB the worst, accuracy is not the best metric to evaluate the performance. It is interesting that, in scenarios like intrusion detection, the algorithms are biased toward estimating all the samples as normal. Even if an algorithm detects all the samples (even the attack ones) as normal, the accuracy will still be high, since the attack samples consist of a very small part of the dataset. This is a major problem that comes with imbalanced dataset. We will discuss this a bit further in the next subsection.

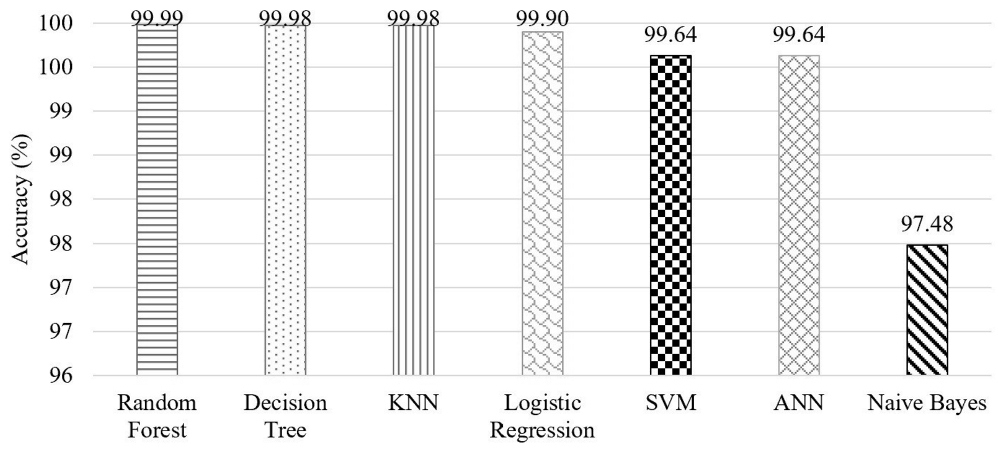


Figure 3.4: Accuracy

The false alarm rate (FAR), shown in Figure 3.5, represents the percentage of the normal traffic being misclassified as the attack traffic by the model (Eq. 2.2). Figure 6 shows good performance for all the models except NB. However, even this metric alone cannot truly represent the performance. Since the number of normal traffic is considerably higher than the attack data, and also the models are biased to label almost all the test data as normal (due to the imbalanced training dataset), the FAR value is expected to be low.

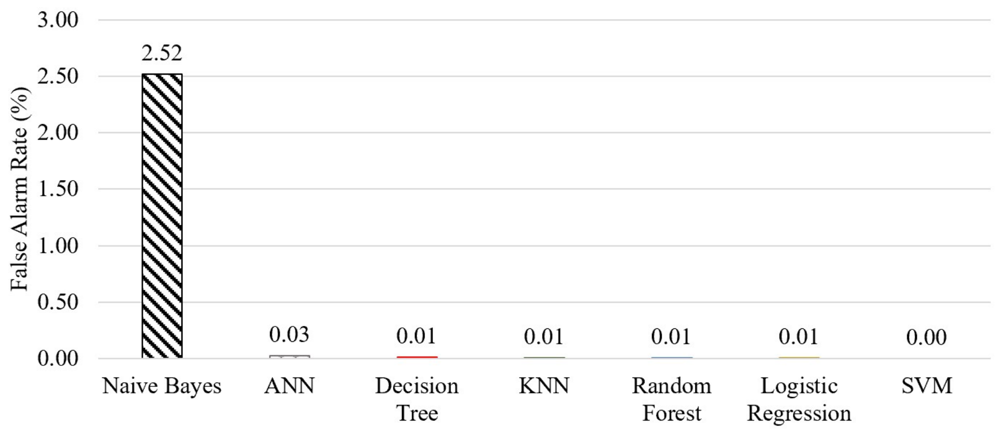


Figure 3.5: False alarm rate

Undetected rate (UR) metric can assess the performance better in spite of the imbalanced data. As shown in Figure 3.6, UR represents the percentage of the attack traffic that is misclassified as normal (the opposite of the FAR) (Eq. 2.3). Since this metric considers only the attack traffic, the fact of having an imbalanced dataset does not impact the evaluation. LR has the worst performance, even compared to a detector that would randomly assign true and false to each traffic packet, which would lead to 50% UR with an infinite number of packets. However, RF showed the best performance. This metric is more critical than FAR because it is related to the attacks not being detected by the system.

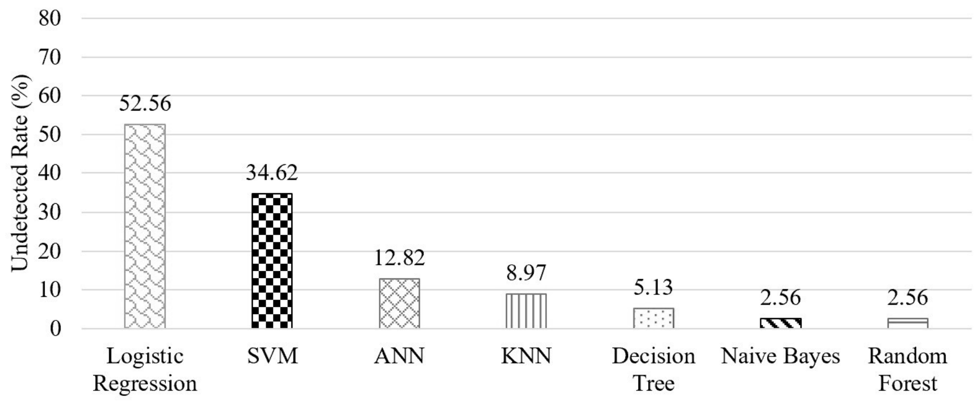


Figure 3.6: Undetected rate

Figure 3.7 shows the ROC (receiver operating characteristic) curve. This curve basically plots the TP rate versus the FP rate for each model. As depicted, while RF shows the best performance, LR has the worst performance. The poor performance of LR for this metric is due to the low TP rate of the model in detecting abnormal traffic.

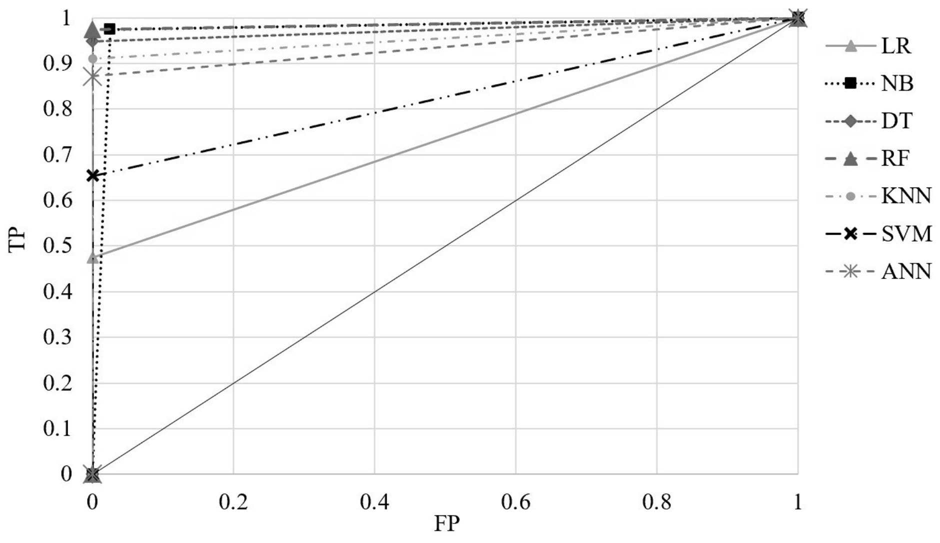


Figure 3.7: ROC curve

MCC (Eq. 2.4), shown in Figure 3.8, is considered to be one of the best metrics for classification evaluation, and it is generally a better performance representative compared to the ROC curve and other metrics. As shown in this figure, RF has the best MCC value, while NB has the worst. MCC is considered as a fair metric when it comes to evaluating ML models that were trained with an imbalanced dataset. Since this metric represents the correlation agreement between the observed values and the predicted values, it is less affected by severe imbalanced ratios.

Finally, the sensitivity metric results (Eq. 2.5) are shown in Figure 3.9 to evaluate how sensitive each model is in reacting to an abnormal situation. As seen in the figure, RF and NB have the highest sensitivity, while LR shows the lowest.

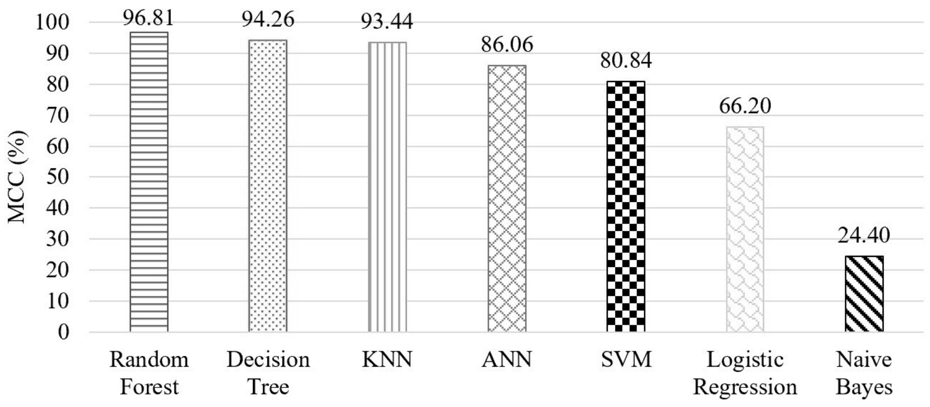


Figure 3.8: MCC

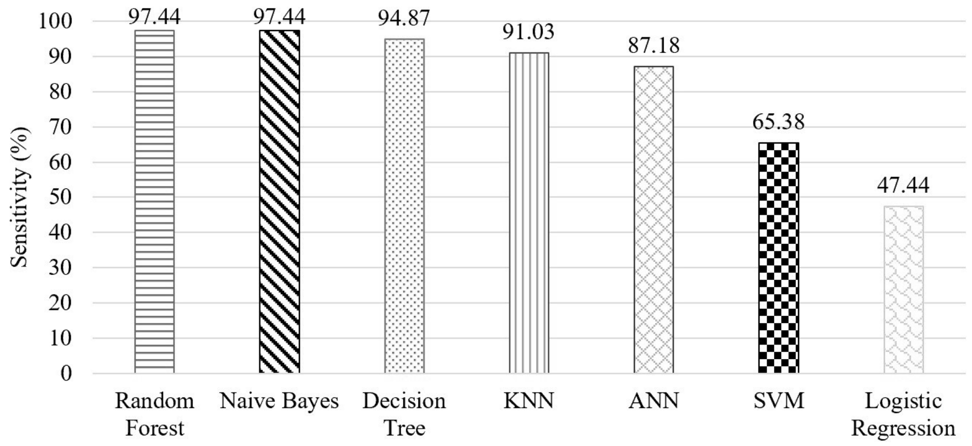


Figure 3.9: Sensitivity

3.9.2 Comparing Different Imbalance Settings

The goal here is to examine the efficiency of a learning model in detecting anomaly through different imbalance ratios. The number of attacks at each trial has been kept equal to 10000 samples, and accordingly, we added normal traffic to build the desired ratios. Table 3.4 is a summary of the number of samples used. At each round of training, we have divided the dataset into 80% for training and 20% for testing.

Table 3.4: Imbalanced settings of our developed dataset

Ratio	**number of Attack**	**number of Normal**	**Total**
10%	10000	90000	100000
1.0%	10000	990000	1000000
0.7%	10000	1418572	1428572
0.3%	10000	3323334	3333334
0.1%	10000	9990000	10000000

To have only one variable (i.e., the imbalanced ratio) in the controlled experiments, we only use command injection attacks as the presented attacks in the datasets. If we include more types of attacks and randomly pick, we might get more of one type in a setting compared to others and it would not be a fair comparison.

Through all these figures, each point on the graph is marked with the corresponding ratio (e.g., "10%" means the ratio of attack samples to the normal samples is 1 to 9). As shown in Figure 3.10, representing the accuracy results (Eq. 2.1), it seems there is not much difference in accuracy performance. However, this is not true. As mentioned before, in scenarios with severe imbalanced ratios, accuracy is not the best representative metric to evaluate the performance. Since a large portion of training data is normal traffic, the algorithms are biased toward estimating all the data as normal and ignoring the small portion of the attack instances.

The FAR metric, as shown in Figure 3.11, represents the percentage of the normal traffic being misclassified as the attack traffic by the model (Eq. 2.2). As again it is seen, Figure 3.11 shows good performance for all the cases. However, for the same reason, even this metric cannot truly represent the performance. For example, Since the amount of attack traffic is

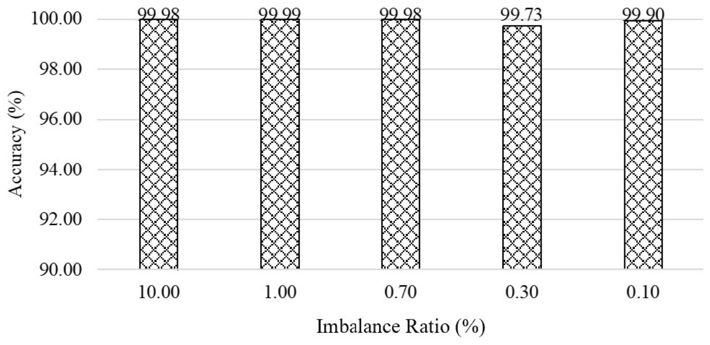


Figure 3.10: Effect of imbalance on accuracy

considerably low in the 0.1% scenario, the algorithms would hardly label any instance as an attack; hence, we would expect a low (even zero) FAR percentage.

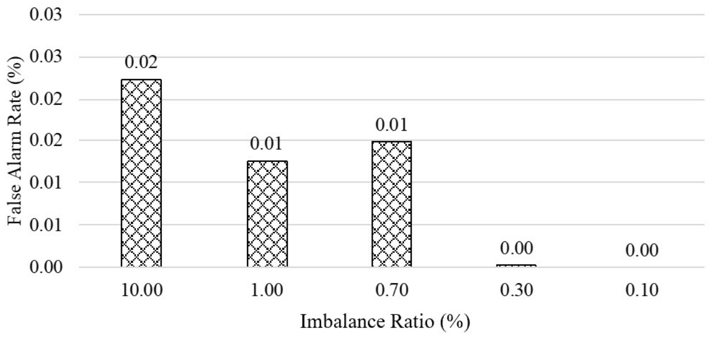


Figure 3.11: Effect of imbalance on false alarm rate

The UR metric can assess the performance better despite having an imbalanced dataset. As shown in Figure 3.12, UR represents the percentage of the attack traffic misclassified as normal (the opposite of the FAR) (Eq. 2.3). Since this metric considers only the attack traffic, the fact of having an imbalanced dataset does not impact the evaluation that much. The training with 0.1% attack was barely able to detect any anomaly and showed the worst performance as it was expected. This metric is more critical than FAR because it is related to the undiscovered attack data.

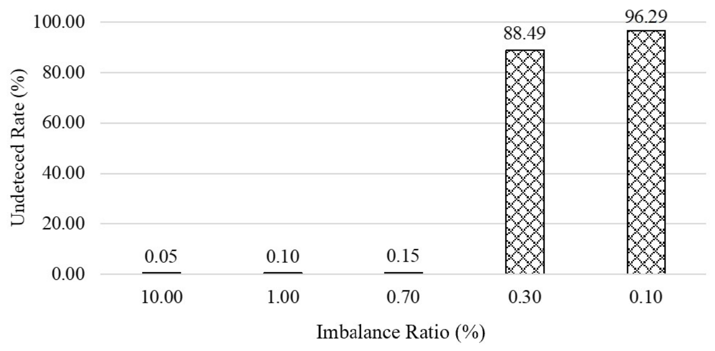


Figure 3.12: Effect of imbalance on undetected rate

The sensitivity metric results (Eq. 2.5) are shown in Figure 3.13 to evaluate how sensitive the model is to react to an abnormal situation. As seen in the figure, training with more abnormal traffic will result in more sensitive detection performance.

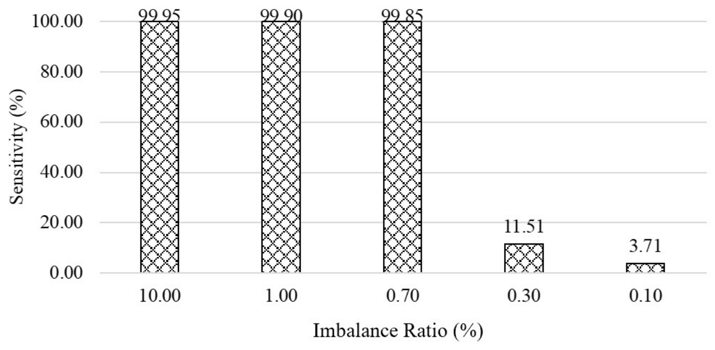


Figure 3.13: Effect of imbalance on sensitivity

Finally, MCC (Eq. 2.4) is considered to be one of the best metrics for classification evaluation. As shown in Figure 3.14, the more attack data is used for training, the better the MCC value we would get. MCC is considered as an appropriate metric when it comes to evaluating ML models trained with an imbalanced dataset.

As mentioned before, we use SMOTE to provide a performance comparison with a baseline sampling method. In this technique, we synthesize new fake attack data from the existing

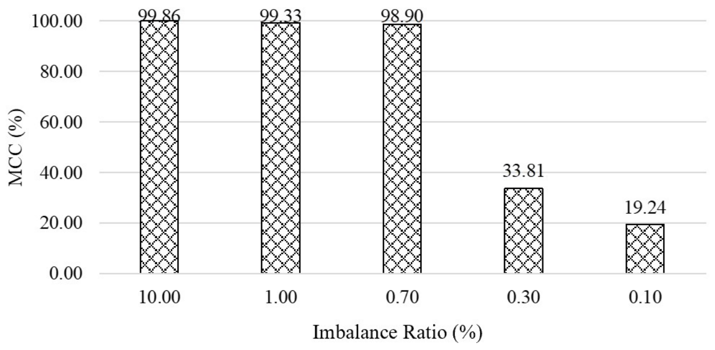


Figure 3.14: Effect of imbalance on MCC

attack samples based on their k nearest neighbors. We ran this method only for 7%, 3%, and 1% anomaly ratios since these three were severe cases.

Figure 3.15 shows the undetected rate before and after using the SMOTE technique. As it is shown in this picture, this method decreased the rate of undetected attacks to 0 for 0.7% and 0.3% imbalance ratios and to about 57% for the 0.1% case. Even though through this technique at 0.1%, we achieved a better rate, still more than half of the attack data were not detected.

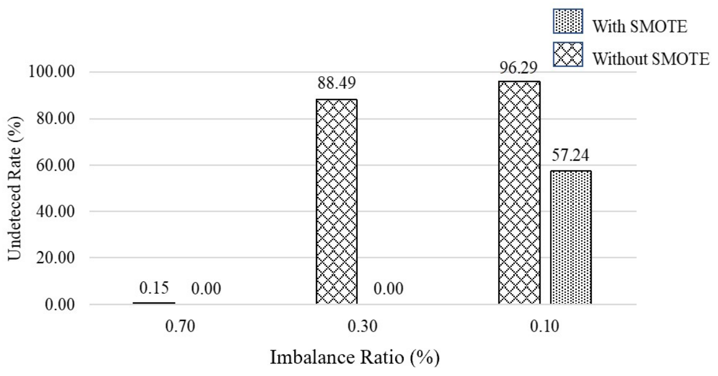


Figure 3.15: Effect of SMOTE method on undetected rate

The MCC results are shown in Figure 3.16. We observe an improvement in the 0.3% and 0.1% cases, with a slight degradation for the 0.7% imbalance ratio. As a result, this oversampling technique helped the system distinguish the attack scenarios better, but still did not perform as good as we wanted in severe imbalanced case.

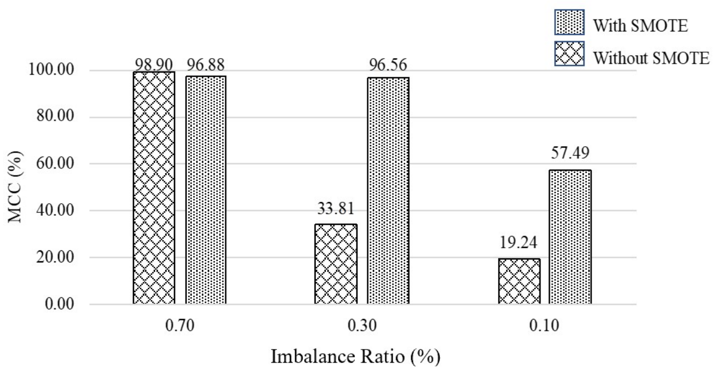


Figure 3.16: Effect of SMOTE Method on MCC

To provide a better judgment for severe cases of imbalanced datasets, we also utilized the k-fold cross validation with $k = 10$ for the 0.1%, 0.3%, and 0.7% ratios after applying SMOTE. Tables 3.5, 3.6, and 3.7 show the UR, sensitivity, and MCC results respectively. As seen, the performance results are highly dependent on the divisions of the training and test sets. The outliers show that an improper division will degrade the performance significantly. These outliers are highlighted in the table. This emphasize the fact that it is crucial to do a cross validation and take the average over all the outcome for a more realistic results.

Table 3.5: UR results of a 10-fold cross validation

UR % for 0.1	UR % for 0.3	UR % for 0.7
90.50	0	0
88.32	0.10	0.098
86.60	0	0.10
88.73	89.06	0
96.62	0.096	0.51
88.42	0.10	0
87.47	0	0.096
88.66	0.10	89.75
89.47	0.19	0
87.47	0.098	0

Table 3.6: Sensitivity results of a 10-fold cross validation

Sensitivity % for 0.1	Sensitivity % for 0.3	Sensitivity % for 0.7
9.49	100	100
11.67	99.89	99.90
13.39	100	99.89
11.26	10.93	100
3.37	99.90	99.48
11.57	99.89	100
12.52	100	99.90
11.33	99.89	10.24
10.52	99.80	100
12.52	99.90	100

Table 3.7: MCC results of a 10-fold cross validation

MCC % for 0.1	MCC % for 0.3	MCC % for 0.7
30.79	99.38	99.65
34.15	98.80	99.50
36.58	99.70	99.04
33.55	33.02	99.70
18.37	98.43	99.58
34.01	99.00	99.44
35.37	99.39	99.27
33.64	99.30	25.53
32.42	99.70	99.66
35.37	98.69	99.74

The performance of both 0.3% and 0.7% cases are somewhat close. This shows even with a 0.3 % ratio, we might get good results if the data set is divided in a way that it is very discriminating training set for the attack and normal samples. On the other hand, it is clear that in all cases, 0.1% ratio scenario performs very poorly.

It is also important to notice that when it comes to the k-fold cross-validation, the algorithm dedicates $k-1$ times more samples to train and less to testing, which might not a very fair division in training problems. It is a known fact that if we divide the dataset so that a lot more samples would be in the training data, the performance results will have greater variance since it might get over-fitted. Further, the trained model might perform very poorly for new unseen samples. That is why a common rule of thumb is the division ratio of 80% for training and 20% for testing.

3.10 Conclusion and Future Direction

The cyber-security of the IIoT devices is critical. Intrusion detection is the main security concern in these applications. Security solutions using learning models and big data analytic have been widely used to ensure a secure platform in these systems. However, when it comes to a real-world scenario and applying these algorithms practically, they sometimes fall short. The main focus of this chapter was studying the scarcity of specific data to IIoT applications. Meanwhile, it is important to take into considerations which attributes separates them from other applications. One this attributes is their imbalanced data problems.

We helped solve the data scarcity for IIoT by building a testbed, running prevalent attacks against it and building datasets. Based on an extensive study, attacks that target the authorization security control have not been investigated in the available research works. Therefore, in addition to other security controls (integrity, availability, confidentiality), we ran backdoor attacks to include authorization threats in our dataset as well.

By releasing our comprehensive, well-studied, and carefully cleansed dataset, we have helped the research community in the field of IIoT's security using AI/ML techniques. This will allow the researchers conducting analysis on a real-world dataset which ultimately paves the way of utilizing AI in practice for these critical infrastructures and being able to exploit modern and up-to-date security solutions. Feature importance ranking was also studied to highlight the most salient features in distinguishing the attack traffic from the normal traffic.

In the evaluation section, we have presented how learning models are capable of filling the identified gap by handling the prevalent attacks. Special attention was also paid to evaluate the performance of the system using better representative metrics.

We showed in which extend the learning algorithms are able to help. When it gets to severe case of 0.1%, even an over-sampling technique would not be able to help. We used SMOTE as a popular technique to synthesize new samples in the minority class (i.e., attack). Further investigation and methods need to be developed to solve the problem of severe cases.

As a future direction, there is a need to focus on utilizing a joint design of multiple algorithms to achieve better performance. This is especially important in the case of severe imbalanced datasets. The hybrid model should be able to provide more accurate results compared to any of the constituent models. False negatives, even a low number of them, mean malicious exertions against the system that stayed undetected and could lead to catastrophic results. Hence, reducing the number of false negatives is what we plan to concentrate on. Ensemble learning is one way to go. They help decrease the variance of the output while improving the performance.

Chapter 4

Black-Box Nature of AI

Despite AI's significant growth, its "black box" nature creates challenges in generating adequate trust. Thus, it is seldom utilized as a standalone unit in IoT high-risk applications, such as critical industrial infrastructures, medical systems, and financial applications, etc. Explainable AI (XAI) has emerged to help with this problem. However, designing appropriately fast and accurate XAI is still challenging, especially in numerical applications. Here, we propose a universal XAI model named Transparency Relying Upon Statistical Theory (TRUST) for XAI, which is model-agnostic, high-performing, and suitable for numerical applications. Simply put, TRUST XAI models the statistical behavior of the AI's outputs in an AI-based system. Factor analysis is used to transform the input features into a new set of latent variables. We use mutual information to rank these variables and pick only the most influential ones on the AI's outputs and call them "representatives" of the classes. Then we use multi-modal Gaussian distributions to determine the likelihood of any new sample belonging to each class. We demonstrate the effectiveness of TRUST in a case study on cybersecurity of the IIoT using three different cybersecurity datasets. As IIoT is a prominent application that deals with numerical data. The results show that TRUST XAI provides explanations for new random samples with an average success rate of 98%. Compared with LIME, a popular XAI model, TRUST is shown to be superior in the context of performance, speed, and the method of explainability. In the end, we also show how TRUST is explained to the user.

4.1 Introduction and Motivation

The impact of AI on today's technological advancements is undeniable. Its applications range from daily life events such as medical decision support to fundamental demands such as cybersecurity. Despite the popularity of AI, it is limited by its current inability to build trust. Researchers and industrial leaders have a hard time explaining the decisions that sophisticated AI algorithms come up with because they (as AI users) cannot fully understand why and how these "black boxes" make their decisions.

Trusting AI blindly impacts its applicability and legitimacy. Based on Gartner's predictions, 85% of the AI projects until 2022 will produce inaccurate outcomes resulting from the organization's limited knowledge of the deployed AI and its behavior [28]. Also, the increasing growth in implementing AI has raised concerns of 91% of cybersecurity professionals about cyberattacks using AI [95]. In the medical field, dedicated AI algorithms are able to precisely (sometimes even more accurately than a human practitioner) detect rare diseases [13]. However, due to the lack of AI's decision transparency and explainability, the results are not considered trustworthy enough to be utilized in practice [37]. A global survey shows that more than 67% of the business leaders do not trust AI, and they believe AI's ambiguity will have negative impacts on their business in the next five years [64].

Moreover, the IoT technology, as an integral part of both personal lives and industrial environments, has profited from AI for AI-powered data analysis. It allows connections, interactions, and data exchange among machines and devices for comprehensive functionality and higher efficiency without requiring any human in the loop. This automation requires a tremendous amount of trust while utilizing AI. This fact emphasizes why incorporating XAI in an AI-based IoT system is essential.

Designing self-explanatory models has gained much attention recently. XAI deals with psychology and cognitive science to provide transparent reasoning for AI's actions [57]. Based on how the underlying AI is designed and the input types (image, text, voice, numerical data), the methods of explainability differ.

A proper XAI model should be integrated into the system to gain trust and transparency for AI-based systems. For clarity, in this dissertation, we call the main underlying AI model the *primary AI model* and the explainable model that accompanies it, the *XAI model*. The primary AI model is in charge of the main functionality of the system, such as classification, regression, recommendation, recognition, etc. The XAI model is in charge of providing transparency and explanation in the primary AI's behavior.

The primary AI and the XAI could be both in one model, which we can interpret the model's outcomes directly using the model itself. This is achievable by using interpretable models as the primary AI model (e.g., decision tree, linear regression, etc.). Even though this type of XAI might provide potentially more accurate explanations, it is limited by the non-reusability, and we can only count on the model providers [2]. Also, learning performance and explainability of learning models usually have an inverse relationship [37]. Better learning performance comes with more sophistication in finding complex relationships in the data features. As such, it becomes more challenging for individuals to grasp its rationality. Therefore, in some applications, choosing interpretable models as the primary AI might yield a significantly degraded learning performance. Hence, surrogate explainers[4] running in parallel along with

[4]This means developing a separate model working in parallel with the primary AI model to explain its behavior.

the primary AI have become preferable in the explanation domain. They are independent and do not affect or put restrictions on the performance of the primary AI model.

It is important to note that, in the XAI domain, it is not expected that the explanation would be in layman's terms. Most of the developed XAI models in the literature are just another interpretable AI or machine learning (ML) model that is easily understood. Therefore, it is assumed that the user is an expert in the internal workings of the learning model utilized as the XAI and knows the mathematics behind it. Here, we argue that statistics are significantly more verifiable and explainable than the AI or ML models used as the XAI. Therefore, we propose using statistical models to simplify the explanation process.

Here, we introduce an XAI model that provides transparency relying upon statistical theory (TRUST). TRUST XAI is a surrogate explainer that provides interpretability without sacrificing the performance of the primary model or imposing restrictions on it. The explainer does not depend on the type of the primary AI in any way; thus, it is entirely model-agnostic[5]. We believe that TRUST XAI will complement the future AI deployments as a standalone unit by providing transparent, independent, and reliable explanations.

While many research works in the XAI domain have focused on image-based data [2, 48], TRUST XAI applies to numerical data such as network data for IoT and security systems, which is critical but currently deficient in XAI.

The TRUST technique works solely with the feature sets and the outputs from the primary AI. If the primary AI is modeled as shown in Figure 4.1 on the left side, TRUST XAI is represented by the three steps on the right using a background with diagonal stripes. The

[5]This simply means the XAI does not care about the type of the primary AI algorithm. All it requires is being able to probe the model with any arbitrary input and get an output from the “black box".

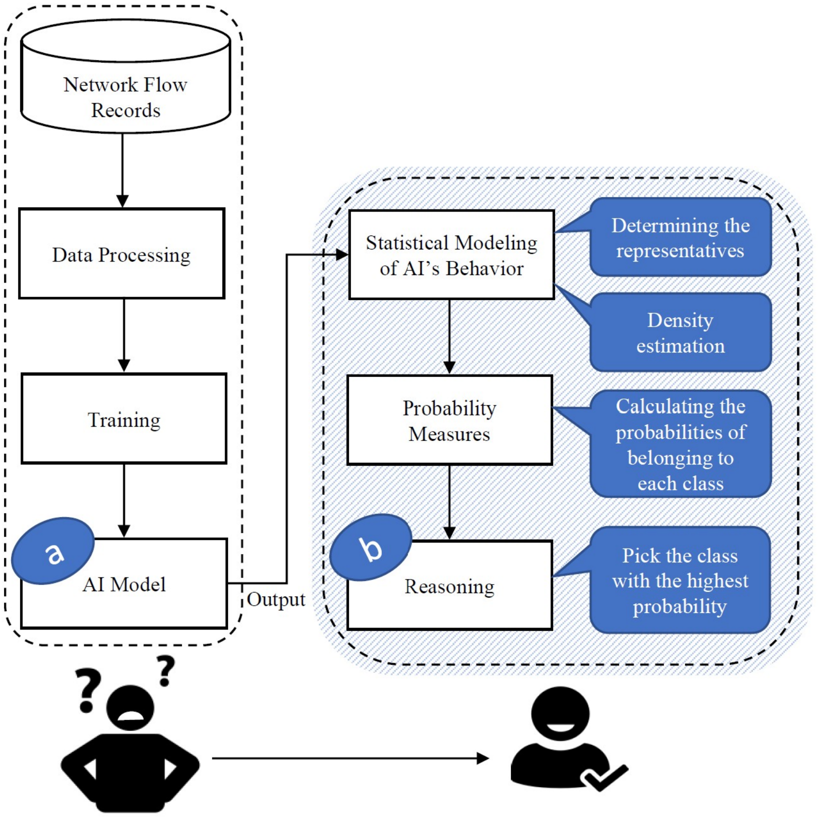

Network Flow Records
Data Processing
Training
AI Model
a
Output
Statistical Modeling of AI's Behavior
Determining the representatives
Density estimation
Probability Measures
Calculating the probabilities of belonging to each class
Reasoning
b
Pick the class with the highest probability
?
?

Figure 4.1: Integrated TRUST explainer in an AI-based system. While (a) can be very complex and non-interpretable, (b) should be as easily interpretable as possible for humans

first step builds the core of the explainer, and it has to be calculated just once. Then, only the last two steps are needed for explaining any new data instances.

To evaluate and provide a practical instance for TRUST XAI, we use three datasets: 1) "WUSTL-IIoT-2021, our dataset, as mentioned in Section 3.7, generated from our IIoT testbed; 2) "NSL-KDD," a public dataset for network-based IDSs from University of New Brunswick [88]; and 3) "UNSW," another IDS dataset from the Cyber Range Lab of the Australian Centre for Cyber Security (ACCS) [61]. We empirically prove that our proposed XAI model successfully explains the labels predicted by the primary AI in just a few milliseconds. Moreover, we have implemented and applied Local Interpretable Model-agnostic Explanations (LIME) on the three mentioned datasets for the sake of comparison. Our results show LIME is much slower than our model and is not practical for real-time applications or when dealing with a large number of samples.

As data privacy and integrity are essential in all areas of IoT, cybersecurity is of paramount importance. Utilizing AI to provide a secure platform in this area has been proven as an effective method, especially in mission-critical applications such as IIoT [18], [6], and [107]. Our case study on the security of IIoT indicates our belief in the importance of IIoT and Industry 4.0. However, TRUST XAI is universal and can be applied to any other applications with minor or zero modification.

The rest of this chapter is organized as follows. The related research papers are explored first. Afterward, we discussed the proposed TRUST model, including how we formalize the problem, what the assumptions are, and how we develop a proper model. Following that, to test TRUST, we thoroughly evaluate its performance in a case study using the three aforementioned datasets. How TRUST would interpret the results to the user is also explained. Finally, we end this chapter with the conclusion section, while mentioning the limitations of TRUST and future directions that can be taken.

4.2 Related Work

The application of XAI in the IoT domain has emerged as a substantial promise for improving trust and transparency. Examples include industrial settings [26], [51], smart grids [81], 5G telecommunication [33], smart homes [27], and healthcare [5], [90], along with XAI surveys covering different other areas, such as [2], [32], [20], [4], [54], and [7]. XAI has been recognized as a necessary trend toward the next generation of AI-based systems, aiming to produce transparent models without affecting the primary AI's accuracy. The trade-offs between accuracy, interpretability, reusability, and performance have been continuously studied [2, 13, 32, 57].

As discussed in [59], feature importance is the most commonly utilized method of explainability. It is the basic technique to find the key features impacting the AI's outcome. Feature selection is a common tool in data preprocessing [47]. It discovers important features that affect the AI's output in general across all classes. However, as seen in the following works, pinpointing these features alone cannot provide a baseline for evaluating or comparing the performance of the proposed XAI.

For example, in [94], the SHAP (SHapley Additive exPlanations) values, which is a feature importance technique originally developed by [49], are used to explain the predictions made by the primary AI model in network security. The results of the XAI model are presented as the list of features that are more indicative of the class of data. However, even the authors confirmed that SHAP is computationally heavy and not suitable for real-time applications. In [9], another feature importance technique, layer-wise relevance propagation (LRP), is developed and specialized for the explainability of deep neural network (DNN) models. It should potentially provide accurate explanations for DNN, but it cannot be generalized for other models.

In [75], modeling a random forest with a decision tree has been proposed to add interpretability to the results. We know that while decision trees are highly explainable, random forest suffers from being complex and lack in interpretability. However, this technique is not model-agnostic since it is only applicable in tree-based learners. In [53], an adversarial technique is used as a means of interpretability. The method calculates the minimum adjustments to the important features' values of a misclassified sample until the primary AI correctly predicts it. However, the proposed method is semi-agnostic since it works only for the AI models with a defined gradient cross-entropy loss function. Additionally, no performance assessment is provided.

Finally, LIME, which is used as a *benchmark XAI* in the performance evaluation, has been a popular technique in the XAI domain, which has many derivatives since its introduction in [73], such as [72, 103]. It generates local surrogate models for every instance to explain its label. For each instance, a new dataset is synthesized consisting of samples with feature values drawn from a normal distribution with a mean and standard deviation corresponding to that instance. The primary AI then labels the synthesized samples. Also, they are assigned weights based on their distance from the instance of interest. Next, a simple interpretable AI or ML model (usually a linear regression model) is trained with the new synthetic dataset to explain the outcome of the primary AI for that specific instance. LIME is re-evaluated and compared with Anchors introduced in [74] by the same authors. Their results show that a linear explanation like LIME is preferred in cases where predictions are near the primary AI's decision boundary because linear explanations provide insight into the primary AI, while Anchors contributes by providing the coverage of the explanations with high accuracy (e.g., when the word "not" has a positive influence on the sentiment, such as "not bad"). Later in this chapter, we compare the performance of LIME with our TRUST model and show how our model outperforms LIME in terms of speed and performance.

4.3 Proposed TRUST Explainer

The goal of TRUST XAI is to explain the rationale behind the data labeling of the primary AI model by applying statistical techniques to the AI's outputs. Table 4.1 presents the symbols used in this chapter. Symbols that are explained along with the equations or only used once are skipped from this table due to length considerations.

Table 4.1: Symbol table

Symbol	**Description**
N	Number of observations in the training set
C	Total number of classes
c	Index indicating class number, $c \in \{1, ..., C\}$
N_c	Number of observations classified by the primary AI as being in class c, $\sum_{c=1}^{C} N_c = N$
K	Number of features used in the primary AI model
k	Number of representatives used in the XAI model, $k \ll K$
i	Index indicating the feature number $i \in \{1, ..., K\}$ for the primary AI model or the representative number $i \in \{1, ..., k\}$ for the XAI model
f_i	i^{th} feature
$\mathcal{F}$	The feature set: $\{f_1, ..., f_K\}$
F_i^c	Random variable of i^{th} factor for class c, $F_i^c \in \mathbb{R}^{N_c \times 1}$
F_i	The i^{th} factor; vertical concatenation of factors $\{F_i^1, \cdots, F_i^C\}$
R_i^c	Random variable of i^{th} representative for class c, $R_i^c \in \mathbb{R}^{N_c \times 1}$
R_i	The i^{th} representative; vertical concatenation of $\{R_i^1, \cdots, R_i^C\}$
$r_{j,i}^c$	R_i^c value in the jth observation, $j \in \{1, ..., N_c\}$
$\boldsymbol{r_i^c}$	Vector of observations belonging to the random variable R_i^c
$\boldsymbol{R^c}$	$N_c \times k$ observation matrix consisting of $r_{j,i}^c$ values arranged so that the column i consists of $\boldsymbol{r_i^c}$'s values
M_i^c	Number of modes of R_i^c in its MMG distribution
w_i	Importance coefficient of R_i
λ_i^c	Eigenvalue of F_i^c

4.3.1 Problem Formalization

To formalize the problem that the primary AI is solving, TRUST XAI produces a set of vectors called "representatives" from the features of the training set. We make three assumptions about the dataset and the representatives as follows.

Assumption 1: The statistical inferences made by the XAI model must represent the data's characteristics. Therefore, the data used to train the primary AI is also used to build the TRUST model. If the primary AI model has to be retrained due to a significant change in the data's characteristics, the core of the explainer must be rebuilt as well.

Assumption 2: The representatives are mutually independent. This allows their joint probability function to be the multiplication of individual probability functions. This assumption simplifies our XAI technique and makes it easier for human users to understand. We select representatives so that the correlation among them is close to zero.

Assumption 3: Suppose for the i^{th} representative R_i, we have a sample $\boldsymbol{r_i^c}$ with N_c observations belonging to the class c. The distribution of these observations can be approximated with reasonable accuracy by a one-dimensional multi-modal Gaussian (MMG) distribution with M_i^c sub-populations, which are called *modes*, with $\{\mu_{i,1}^c, ..., \mu_{i,M_i^c}^c\}$ means and $\{\sigma_{i,1}^c, ..., \sigma_{i,M_i^c}^c\}$ standard deviations. In the case of a unimodal distribution, M_i^c is equal to one.

4.3.2 Determining the Representatives

Since factor analysis [65] satisfies the requirements mentioned in Assumption 2 and more, we use it to determine the representatives. It provides a linear combination of the dataset's features. It also reduces the redundancy in the feature space, which makes our model less likely to overfit. Further, since these new independent latent variables are combinations of the features, we still preserve the most valuable parts of all features. Factor analysis projects the feature values onto axes that maximize the percentage of explained variance in the data. Another advantage of this technique is that the resulting factors are orthogonal. Therefore, they are independent with zero covariations.

More specifically, we use factor analysis of mixed data (FAMD) that takes into account different types of variables (quantitative and qualitative) in a dataset [65]. FAMD is considered the general form of factor analysis. Generally speaking, the core of FAMD is based on principal component analysis (PCA) when variables are all quantitative and multiple correspondence analysis (MCA) when all the variables are qualitative.

When dealing with a mixed dataset, the typical approach in the research community is either neglecting or encoding (e.g., one-hot encoding) the categorical data when using PCA or transforming the quantitative variables into qualitative variables by breaking down their variation intervals into different levels when using MCA. Despite being relatively easy to implement, these approaches are not accurate enough, and we lose lots of information from the data. Therefore, FAMD is considered the most accurate form of analysis when dealing with a mixed dataset because it takes into account both types of data [42].

Suppose we have a feature set $\mathcal{F} = \{f_1, ..., f_K\}$ of mixed data, where K is the total number of the features. After applying factor analysis to the observations belonging to each class separately, we have K factors determined for each class. Note that the number of features and factors are both K. For example, for class c, factors $\{F_1^c, ..., F_K^c\}$ are produced. The pseudo-code of this function, `factorAnalysis`, is presented as Algorithm 1. In this algorithm, each data instance belongs to one of the C classes.

In Algorithm 1, $\rho^2_{f_i,f_j}$ is the Pearson correlation coefficient, $\chi^2_{f_i,f_j}$ is the chi-square, and $\eta^2_{f_i,f_j}$ is the squared correlation ratio between feature i and feature j from the standardized matrix of X^c [39, 55]. When measuring the relationship between two quantitative variables Pearson correlation coefficient is used, and for two qualitative variables chi-square is used. Further, when dealing with one variable from each kind, we use the general form of the squared correlation ratio, which results in the Pearson correlation coefficient if both variables are quantitative or chi-square if they are both qualitative. Simply put, these three methods all boil down to the same result, measuring correlations between each pair of variables building a relation matrix for each class. Lastly, singular value decomposition (SVD) is applied to the relation matrix RM^c for factorization, removing the interdependencies among the features and projecting them in orthogonal dimensions.

Algorithm 1 Factor Analysis

```
Function: factorAnalysis
Input: Training Set: $X \neq \emptyset \in \mathbb{R}^{N \times K}$;
    AI Model: $\mathcal{AI}$; Features Set: $\mathcal{F}$
 1: $y \leftarrow \mathcal{AI}(X)$
 2: Set labels of each row in $X \leftarrow y$
 3: Divide $X$ into $X^c \in \mathbb{R}^{N_c \times K}$ sets per class
 4: for each $X^c$ do
 5:     $X^{c\prime} \leftarrow$ standardized $(X^c)$ ▷ So it has zero mean and unit variance
 6:     Build the relationship matrix $(RM^c)$ such that:
 7:     for each pair $(i, j)$ in $\mathcal{F}(X^c)$ do
 8:         if $f_i$ & $f_j$ are both quantitative then
 9:             $RM^c_{i,j} = \rho^2_{f_i,f_j}$
10:         else if $f_i$ & $f_j$ are both qualitative then
11:             $RM^c_{i,j} = \chi^2_{f_i,f_j}$
12:         else
13:             $RM^c_{i,j} = \eta^2_{f_i,f_j}$
14:         end if
15:     end for
16:     $\{F^{c\prime}_1, ..., F^{c\prime}_K\}$ & $\{\lambda^c_1, ..., \lambda^c_K\} \leftarrow$ SVD$(RM^c)$
17:     $\{F^c_1, ..., F^c_K\} \leftarrow$ unstandardized $(\{F^{c\prime}_1, ..., F^{c\prime}_K\})$
18:     return $\{F^c_1, ..., F^c_K\}$
19: end for
```

Then, in Algorithm 2, we use the factors calculated from Algorithm 1 to pick the “representatives" based on how important each factor is.

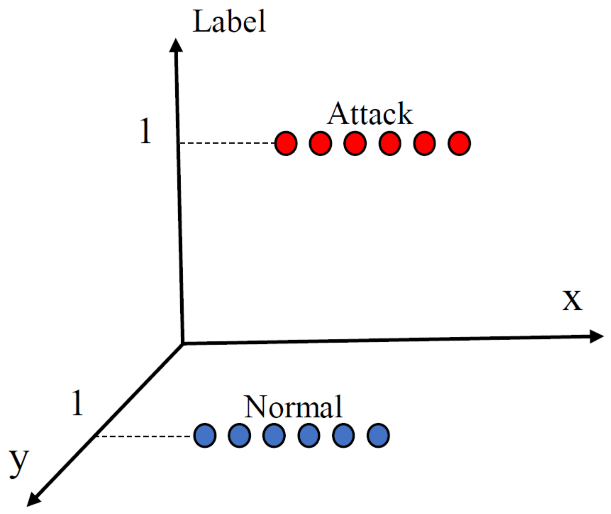


Figure 4.2: Two-dimensional data with features highly correlated with the labels

Factor analysis (Algorithm 1) produces an eigenvalue λ_i^c for each factor F_i^c, which is equivalent to the variation in the data that the factor explains. However, the effectiveness of a factor in distinguishing the class label is not related to the percentage of the explained variation of the data by that factor. Therefore, the percentages of explained variation cannot be used as the importance coefficients for the factors. We elaborate more on this in the following.

Suppose we have two-dimensional data with their class labels on the third axis. An example is demonstrated in Fig. 4.2. The attack class is shown by the red points and the normal class by the blue points. As seen in this figure, since the variation in the y-axis values per class is not large, the y-axis feature would not be considered important. Therefore, in the case of a dimension reduction or factor analysis, separately for each class, the x-axis values would be the primary axis representing each class's samples. This is because x-axis explains almost all the variations in the data points for each class. Meanwhile, it is trivial to see that their y-axis values could easily distinguish the class labels. Hence, removing them or treating them as not important will cause a significant loss in the accuracy of the proposed model.

Therefore, we devised a method using the correlation between each factor and the class label to rank the factors and pick the ones with the highest correlation. We use mutual information (MI), which quantifies the amount of information we can get about one variable by knowing the other. It measures the mutual dependence between two variables, which in our case are the factors and the class labels. MI between the factors and the class labels is calculated as

$$\mathrm{MI}(y, F_i) = \mathrm{H}(y) - \mathrm{H}(y|F_i) \tag{4.1}$$

where $F_i = [F_i^{1^T} \ldots F_i^{C^T}]^T$ is the vertical concatenation of the ith factor values for all classes. The T superscript denotes a transpose of the vector/matrix. $\mathrm{MI}(y, F_i)$ is the mutual information between the class label vectors y and F_i. $\mathrm{H}(.)$ is the entropy function, which is defined as follows.

$$\mathrm{H}(y) = -\sum_c P(y = c) \log(P(y = c)) \tag{4.2}$$

For the sake of simplicity and not having to use differential entropy (since F_i is not discrete), we bin the values in F_i, and ζ_i's are the possible outcomes of the binned random variable F_i. Suppose $P(\zeta_i)$ is the probability of $F_i = \zeta_i$. The conditional entropy $\mathrm{H}(y|F_i)$ is calculated as follows.

$$
\begin{aligned}
&\mathrm{H}(y|F_i) = \\
&\sum_{\zeta_i} P(F_i = \zeta_i)\mathrm{H}(y|F_i = \zeta_i) = \\
&-\sum_{\zeta_i} P(F_i = \zeta_i) \sum_{c} P(y = c|F_i = \zeta_i) \log P(y = c|F_i = \zeta_i) \\
&= -\sum_{\zeta_i}\sum_{c} P(c, \zeta_i) \log \frac{P(c, \zeta_i)}{P(\zeta_i)}
\end{aligned}
\tag{4.3}
$$

We use the MI values as the importance coefficients for the factors. For instance, w_i, the importance coefficient of F_i, is equal to $\mathrm{MI}(y, F_i)$. For more details about MI, we refer to [19].

Then, we pick only the first k factors, $k \ll K$, with the highest w_i values as the "representatives." This means the representatives are the factors that have the highest correlation with the class labels. They represent each class's feature values and behavior.

There is no need to represent each class by its all K factors. Based on the application constraints, we can choose k. Regarding the choice of k (i.e., the number of representatives), there is a trade-off between time consumption and performance. For real-time applications, we can pick a lower k (meaning a few factors with the highest importance coefficients as the representatives). For applications where high accuracy is important, we can go with a higher k. Since the density and mode computations take time, the more representatives we have, the slower the model would be, but at the same time, it leads to higher accuracy. The time consumption analysis is provided later in this chapter. The algorithm to select the representatives is summarized in Algorithm 2.

Algorithm 2 Picking Representatives

```
Function: pickingReps
Input: Training Set: $X \neq \emptyset \in \mathbb{R}^{N \times K}$;
    AI Model: $\mathcal{AI}$; Features Set: $\mathcal{F}$
1: $\{F_1^c, ..., F_K^c\} \leftarrow$ factorAnalysis$(X, \mathcal{AI}, \mathcal{F})$
2: $y \leftarrow \mathcal{AI}(X)$
3: for each $i$ do
4:     $F_i = [F_i^{1^T}, \cdots, F_i^{C^T}]^T$
5:     $w_i \leftarrow \mathrm{MI}(y, F_i)$
6:     $w_i' \leftarrow$ sort $w_i$ in descending order
7:     $\{R_1^c, ..., R_k^c\} \leftarrow$ Pick top $k$ factors with the highest $w_i'$s
8:     return $\{R_1^c, ..., R_k^c\}$
9: end for
```

4.3.3 Density Estimation

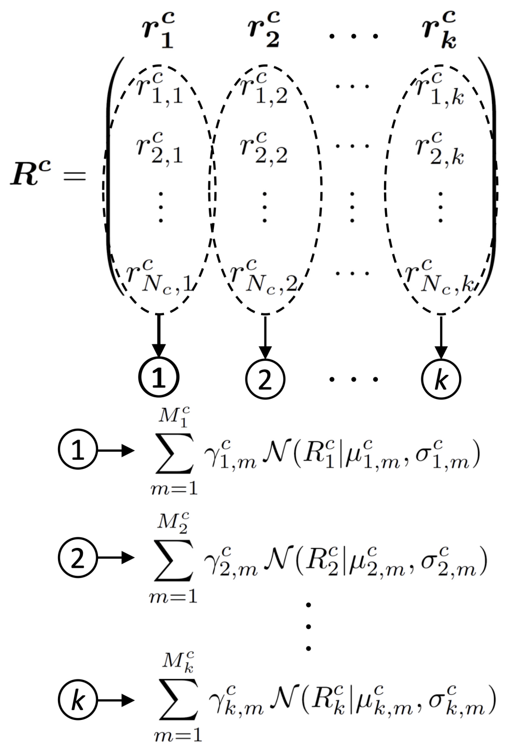
To study the statistical attributes of the representatives' values, we approximate their density functions with one-dimensional MMG distributions.

Suppose for all the data instances of the training set labeled as the class c by the primary AI model, X^c, representative R_i^c $i = 1, .., k$ has M_i^c modes. For now, assume we know the value of M_i^c; we will discuss how to calculate it later in the next section. The one-dimensional multi-modal probability density functions (pdf) of R_i^c can be modeled as follows.

$$p_i(R_i^c) = \sum_{m=1}^{M_i^c} \gamma_{i,m}^c \mathcal{N}(R_i^c | \mu_{i,m}^c, \sigma_{i,m}^c) \tag{4.4}$$

$\gamma_{i,m}^c$ is the component weight of sub-population m for R_i^c observations in class c. $\mathcal{N}$ is a normal distribution with mean $\mu_{i,m}^c$ and standard deviation $\sigma_{i,m}^c$. To make the total area under the pdf normalized for every representative per class, for $\forall c \vee \forall i$, $\sum_m \gamma_{i,m}^c = 1$ should be satisfied. Here we use the expectation-maximization (EM) algorithm [63] to fit the representatives' values to a proper distribution in the form of Eq. 4.4.

After the above process, we end up with $k \times C$ distributions. These distributions build the backbone of the TRUST explainer. Figure 4.3 shows the approximation process for the values of class c representatives as an example. The $\boldsymbol{R^c} = [\boldsymbol{r^c_1}, \boldsymbol{r^c_2}, ..., \boldsymbol{r^c_k}] \in \mathbb{R}^{N_c \times k}$ equates to the matrix of observations of the representatives of the class c, where R^c_i is the random variable for the set of observations $\boldsymbol{r^c_i}$.



$\boldsymbol{r^c_1}$ $\boldsymbol{r^c_2}$ $\cdots$ $\boldsymbol{r^c_k}$
$\boldsymbol{R^c} =$
$r^c_{1,1}$ $r^c_{1,2}$ $\cdots$ $r^c_{1,k}$
$r^c_{2,1}$ $r^c_{2,2}$ $\cdots$ $r^c_{2,k}$
$r^c_{N_c,1}$ $r^c_{N_c,2}$ $\cdots$ $r^c_{N_c,k}$
1 2 $\cdots$ k
1 $\sum_{m=1}^{M^c_1} \gamma^c_{1,m} \mathcal{N}(R^c_1|\mu^c_{1,m}, \sigma^c_{1,m})$
2 $\sum_{m=1}^{M^c_2} \gamma^c_{2,m} \mathcal{N}(R^c_2|\mu^c_{2,m}, \sigma^c_{2,m})$
k $\sum_{m=1}^{M^c_k} \gamma^c_{k,m} \mathcal{N}(R^c_k|\mu^c_{k,m}, \sigma^c_{k,m})$

Figure 4.3: Approximating representatives to unique MMG distributions for class c

Now, it is time to show how our TRUST XAI explains the data instances labeled as a specific class by the primary AI model. First, the likelihoods that the new instance's representatives belong to each class's representatives are calculated. These likelihoods can be estimated from the pdf models. The total likelihood is the product of the likelihoods of all representatives. We apply a logarithm transformation to decrease the computational complexity and compute the log-likelihoods. Therefore, we simplify the computations from multiplications to additions (which is less expensive), avoid computation of the exponentials, and prevent any arithmetic underflow or overflow (since likelihoods could become very large or small). Log transformation also reduces or removes the skewness in the data. Suppose the new instance is $\boldsymbol{z} = [z_1, ..., z_K] \in \mathbb{R}^{1\times K}$. The inputs to TRUST XAI are only the representatives' values. Therefore, $\boldsymbol{z}$ is projected on the factors' space and cropped to its corresponding representatives' values $\boldsymbol{z'} = [z'_1, ..., z'_k] \in \mathbb{R}^{1\times k}$. The conditional likelihood of z'_i, $i = 1, .., k$ belonging to class c can be written as:

$$\begin{aligned} & p_i(z'_i|c) = \\ & \sum_{m=1}^{M_i^c} \exp\Big(\log(\gamma^c_{i,m}) + \log(\mathcal{N}(z'_i|\mu^c_{i,m}, \sigma^c_{i,m}))\Big) \end{aligned} \tag{4.5}$$

After taking a logarithm from Eq. 4.5, the conditional log-likelihood of z'_i belonging to R^c_i is calculated by Eq. 4.6.

$$\begin{aligned} & \log(p(z'_i|c)) = \\ & \log\Big(\sum_{m=1}^{M_i^c} \exp\Big(\log(\gamma^c_{i,m}) + \log(\mathcal{N}(z'_i|\mu^c_{i,m}, \sigma^c_{i,m}))\Big)\Big) = \\ & \log\Big(\sum_{m=1}^{M_i^c} \exp\Big(\alpha^c_{i,m} - \frac{1}{2}(\frac{z'_i - \mu^c_{i,m}}{\sigma^c_{i,m}})^2\Big)\Big) \end{aligned} \tag{4.6}$$

where,

$$\alpha^c_{i,m} = \log(\gamma^c_{i,m}) - \log(\sigma^c_{i,m}) - \frac{1}{2}\log(2\pi) \tag{4.7}$$

$\alpha^c_{i,m}, \forall c \vee \forall i \vee \forall m$ is a fixed constant.

Subsequently, we calculate the weighted sum of these log-likelihoods using the representatives' importance coefficients (the MI values, w_i, from Subsection B). The w_i's are normalized before the calculation, though. The importance coefficient of the representative R_i is calculated by:

$$\hat{w_i} = \frac{w_i}{\sum_{j=1}^{k} w_j} \tag{4.8}$$

Since the w_i's are normalized, $\sum_{i=1}^{k} \hat{w_i} = 1$. After that, the log-likelihood of classifying $\boldsymbol{z'}$ as class c is computed as:

$$\log(p(\boldsymbol{z'}|c)) = \sum_{i=1}^{k} \hat{w_i} \times \log(p_i(z'_i|c)) \tag{4.9}$$

Finally, the predicted class for z by the primary AI, represented as ℓ_z, is explained by TRUST XAI as the class with the maximum total log-likelihood:

$$\ell_{\boldsymbol{z}} = \max_{c} \log(p(\boldsymbol{z'}|c)) \tag{4.10}$$

It is essential to notice that we are not following the common clustering processes that model the whole training set as a multi-dimensional (with features as the dimensions) multi-cluster distribution and then calculate the likelihood of the new data belonging to each cluster (i.e., class). Here, we take a completely different approach. We approximate the one-dimensional distribution of each representative and then calculate the likelihood of the new instance's representative values belonging to the corresponding representative distributions for each class. Afterward, by the weighted sum of these likelihoods for each class, we have C likelihoods, each resulting from k distributions. Then, the class with the highest likelihood is picked as the explained class.

4.3.4 Selecting the Number of Modes

A critical parameter in the TRUST explainer is the number of modes for MMG distribution (i.e., M_i^c) for each representative (as mentioned in Assumption 3). In general, this number is different for each representative, and it can be optimized to increase speed and performance. We use the grid search technique which is a standard method for determining hyperparameters in machine learning techniques. Through grid search, for each representative, we find the number of modes that maximizes the probability of the correct prediction for samples from different classes. The choice of M_i^c for each representative is important since it has a critical role in how well the fitted MMG distribution works as a density estimator. The pseudo-code of the method is shown in Algorithm 3. This algorithm searches over a zone called Z, which is simply a grid of integers starting from 1 to any potential number of modes.

As seen in Algorithm 3, the dimension of the search space grows exponentially with the number of classes. Also, if the number of modes or classes is large, the search zone would be too large. Hence, the search could be very slow and time-consuming. We have developed a faster algorithm that divides the search zone into smaller sub-zones, as is shown in Algorithm 4. At first, the probabilities of the centers of each sub-zone as the number of modes are calculated. Then the sub-zone with the highest score is chosen to be searched thoroughly. Even though this algorithm might not give us the exact number of modes, it provides a good accuracy (in the order of 99% in our experiments). With decreasing the size of sub-zones, the accuracy increases.

Algorithm 3 Grid Mode Selection

```
Function: modeSelect
Input: Representatives of all the classes:
    $\{R_1^1, ..., R_k^1\} \in \mathbb{R}^{N_1 \times k}, \cdots, \{R_1^C, ..., R_k^C\} \in \mathbb{R}^{N_C \times k}$;
    Zone $Z \in \mathbb{R}^{C \times C}$
 1: $y \leftarrow [\text{ones}(N_1)^T, \ldots, C \times \text{ones}(N_C)^T]^T$
 2: for each set of $(R_i^1 \cdots R_i^C)$ do
 3:     $Max \leftarrow 0$
 4:     for $m^1$ in $Z$ do
 5:         $\text{MMG}_i^1 \leftarrow$ fit $R_i^1$ to a Gaussian with $m^1$ modes
 6:         $\vdots$
 7:         for $m^C$ in $Z$ do
 8:             $\text{MMG}_i^C \leftarrow$ fit $R_i^C$ to a Gaussian with $m^C$ modes
 9:             $P_{m^1} \leftarrow p_1(R_i|\text{MMG}_i^1)$
10:             $\vdots$                                    ▷ $R_i = [R_i^{1T}, \ldots, R_i^{CT}]^T$
11:             $P_{m^C} \leftarrow p_C(R_i|\text{MMG}_i^C)$
12:             $P \leftarrow [P_{m^1} \cdots P_{m^C}]$
13:             $y_{\text{MMG}} \leftarrow \text{index}(P.\max(1))$
14:             $\text{score}_{m^1,\cdots,m^C} \leftarrow \text{MCC}(y_{\text{MMG}}, y)$
15:             if $\text{score}_{m^1,\cdots,m^C} > Max$ then
16:                 $Max \leftarrow \text{score}_{m^1,\cdots,m^C}$
17:                 $M_i^1 \leftarrow m^1$
18:                 $\vdots$
19:                 $M_i^C \leftarrow m^C$
20:             end if
21:         end for
22:         $\vdots$
23:     end for
24: end for
25: return $\{M_1^1, ..., M_k^1\} \cdots \{M_1^C, ..., M_k^C\}$
```

As seen in line 14 of Algorithm 3 and line 15 of Algorithm 4, Matthew's correlation coefficient (MCC), Eq. 2.4, is used to calculate the score. We have not used the Accuracy metric, Eq. 2.1, because MCC has been shown to be a better metric when dealing with imbalanced datasets, which are common in cybersecurity and other numerical applications [108]. The undetected rate (UR), Eq. 2.3, is another useful metric for performance evaluation used in the next section.

Algorithm 4 Fast Grid Search

```
Function: modeSelectFast
Input: Representatives of all the classes:
    $\{R_1^1, ..., R_k^1\} \in \mathbb{R}^{N_1 \times k}, \cdots, \{R_1^C, ..., R_k^C\} \in \mathbb{R}^{N_C \times k}$;
    Zone $Z \in \mathbb{R}^{C \times C}$
 1: $y \leftarrow [\text{ones}(N_1)^T, \ldots, C \times \text{ones}(N_C)^T]^T$
 2: for each set of $(R_i^1 \cdots R_i^C)$ do
 3:     Divide zone $Z$ into $z_1, ..., z_d$
 4:     $Max \leftarrow 0$
 5:     for each $z_j$ do
 6:         $(m^1, \cdots, m^C) \leftarrow z_j$'s center
 7:         $\text{MMG}_i^1 \leftarrow$ fit $R_i^1$ to a Gaussian with $m^1$ modes
 8:         $\vdots$
 9:         $\text{MMG}_i^C \leftarrow$ fit $R_i^C$ to a Gaussian with $m^C$ modes
10:         $P_{m^1} \leftarrow p_1(R_i | \text{MMG}_i^1)$
11:         $\vdots$                                          ▷ $R_i = [{R_i^1}^T, \ldots, {R_i^C}^T]^T$
12:         $P_{m^C} \leftarrow p_C(R_i | \text{MMG}_i^C)$
13:         $P \leftarrow [P_{m^1} \cdots P_{m^C}]$
14:         $y_{\text{MMG}} \leftarrow \text{index}(P.\max(1))$
15:         $\text{score}_{m^1, \cdots, m^C} \leftarrow \text{MCC}(y_{\text{MMG}}, y)$
16:         if $\text{score}_{m^1, \cdots, m^C} > Max$ then
17:             $Max \leftarrow \text{score}_{m^1, \cdots, m^C}$
18:             $Z_i \leftarrow z_j$
19:         end if
20:     end for
21:     modeSelect($\{R_i^1\}, \cdots, \{R_i^C\}, Z_i$)
22: end for
23: return $\{M_1^1, ..., M_k^1\} \cdots \{M_1^C, ..., M_k^C\}$
```

4.4 Case Study: Transparent AI for IIoT Security

Sophisticated AI models have been vastly applied in IDS. XAI would add interpretability and trust to these systems. As mentioned previously, we have built a lab-scaled IIoT system to collect realistic and up-to-date datasets for network-based IDSs. We have implemented a SCADA system widely used by industries to supervise the level and turbidity of liquid storage tanks. This IIoT system is employed in industrial reservoirs and water distribution systems as a part of the water treatment and distribution. For more information regarding our testbed, we refer the readers to our papers [107] and [89].

Table 4.2: Specifics of the three datasets

Dataset	WUSTL-IIoT	NSL-KDD	UNSW
# of observations	1,194,464	125,973	65,535
# of features	41	40	41
# of attacks	87,016	58,630	45,015
# of normals	1,107,448	67,343	20,520

4.4.1 Utilized Datasets

The first tested dataset is the one collected from our testbed, called “WUSTL-IIoT-2021," which we refer to as “WUSTL-IIoT." To collect the proper dataset, we (as a white-hat attacker) attacked our testbed using manipulated commands, such as backdoor, command injection, denial of service, and reconnaissance [107].

Further, we have tested our TRUST XAI on two other cybersecurity datasets to add an empirical proof of concept, “NSL-KDD" [88] and “UNSW" [61]. Specifications of these datasets are shown in Table 4.2. NSL-KDD is a revised and cleansed version of the KDD'99 published by the Canadian Institute for Cybersecurity from the University of New Brunswick. KDD'99 was initially collected by MIT Lincoln Laboratory through a competition to collect traffic records and build a network intrusion detector. This dataset consists of four different types of attacks. Here, we label them all as class 1 for binary classification. Further, the UNSW-NB15 dataset, which we refer to as UNSW, was built by the Cyber Range Lab of the Australian Centre for Cyber Security (ACCS) in 2015. The traffics are labeled as either 0 or 1 for normal or attack class, respectively.

4.4.2 The Primary AI Model

TRUST XAI is a model-agnostic technique that can be applied to any complex AI model. To evaluate its performance on IIoT security, we use an artificial neural network (ANN) as a binary classifier with two hidden layers with 20 and 10 neurons respectively with *ReLU* activation functions, ten epochs, with *adam* optimizer, and *sigmoid* activation function in the output layer. The scikit-learn library [67] was used to implement the ANN model (as the primary AI model). ANN is generally considered complex and unexplainable to human users. The inputs to the IDS are the flow instances as mentioned in the previous subsection. The output of the IDS can be a multi-class or binary classification. However, for the sake of simplicity, we have built the TRUST explainer by treating the outputs of the IDS as binary classes with normal as 0 and attack as 1.

Even though the performance of the primary AI is not of our interest, we show its results on the three datasets in Tables 4.3, 4.4, and 4.5 for comparison. Each dataset is split into training and test sets with an 80:20 ratio. The performance on the training set is presented in the (a) tables, while the test results are shown in the (b) tables.

Table 4.3: Performance of the primary AI on WUSTL-IIoT

	Normal	Attack
Normal	885980	12
Attack	131	69448

(a) On the training set

	Normal	Attack
Normal	221452	4
Attack	40	17397

(b) On the test set

Table 4.4: Performance of the primary AI on NSL-KDD

	Normal	Attack
Normal	53664	173
Attack	529	46412

(a) On the training set

	Normal	Attack
Normal	13452	54
Attack	138	11551

(b) On the test set

Table 4.5: Performance of the primary AI on UNSW

	Normal	Attack
Normal	15791	657
Attack	473	35507

(a) On the training set

	Normal	Attack
Normal	3895	177
Attack	115	8920

(b) On the test set

These tables have been brought up to emphasize once again that to develop an XAI in general, only the labels outputted by the primary AI are important while ignoring the true labels of the samples. This is because, in the XAI domain, we care about explaining the primary AI's behavior, not how accurately it has behaved.

The results of these tables based on the metrics including accuracy, MCC, and undetected rate using Eq. 2.1, Eq. 2.4, and Eq. 2.3 are summarized in Table 5.5.

Table 4.6: Summary of the primary AI's performance

	Accuracy	**MCC**	**UR**
WUSTL-IIoT, Training	99.98%	99.88%	0.19%
WUSTL-IIoT, Test	99.98%	99.86%	0.23%
NSL-KDD, Training	99.30%	98.60%	1.13%
NSL-KDD, Test	99.24%	98.47%	1.18%
UNSW, Training	97.84%	94.98%	1.3%
UNSW, Test	97.77%	94.78%	1.27%

Next, we discuss the steps to build the core of TRUST from the output labels from the primary AI. These steps include picking representatives and mode selection.

4.4.3 The XAI Model: TRUST

The first step is to run the `pickingReps` function using Algorithm 2 on the training set with their predicted labels by the primary AI. We end up with K factors. Afterward, the MI scores between each representative and the class labels have been calculated to determine the top k representatives that have the highest correlation with the class labels and their importance coefficients.

As mentioned before, the choice of k provides a tradeoff between the speed and the performance of the explainer. Large k may result in a better performance with a lower speed and vice-versa. Here, just for the sake of experiment, we try k from 1 to 8. Please note that k can be any number between 1 and K. As shown later, we get high MCC scores even with k equal to 4 or 5. Therefore, there is no need to sacrifice the speed or complexity of the model by choosing a large k.

The next step is to estimate the number of modes per representative's probability density function. The general case with the C number of classes has been demonstrated in Algorithm 3. In our case, we have only two classes, $C = 2$, therefore we have only two sets of representatives, $\{R_1^1, ..., R_k^1\}$ and $\{R_1^2, ..., R_k^2\}$. The modified mode computation algorithms for two classes are shown in Algorithm 5 and Algorithm 6.

4.4.4 Results

The representatives of TRUST XAI are built using the training set. The test samples' factors are calculated by projecting their feature values in the factor space computed using the training set. Then the same factors chosen as representatives in the training set are chosen for the test samples as well. Afterward, the likelihoods are calculated for the corresponding

Algorithm 5 Grid Mode Selection with $C = 2$

```
Function: modeSelectC2
Input: Representatives of class "1": $\{R_1^1, ..., R_k^1\} \in \mathbb{R}^{N_1 \times k}$;
    Representatives of class "2": $\{R_1^2, ..., R_k^2\} \in \mathbb{R}^{N_2 \times k}$;
    Zone $Z \in \mathbb{R}^{2 \times 2}$
 1: for each $(R_i^1, R_i^2)$ do
 2:     $Max \leftarrow 0$
 3:     for $m^1$ in $Z$ do
 4:         $\text{MMG}_i^1 \leftarrow$ fit $R_i^1$ to a guassian with $m^1$ modes
 5:         for $m^2$ in $Z$ do
 6:             $\text{MMG}_i^2 \leftarrow$ fit $R_i^2$ to a guassian with $m^2$ modes
 7:             $P_{m^1} \leftarrow p_1(R_i|\text{MMG}_i^1)$        $\triangleright R_i = \begin{bmatrix} R_i^1 \\ R_i^2 \end{bmatrix}$
 8:             $P_{m^2} \leftarrow p_2(R_i|\text{MMG}_i^2)$
 9:             $y_{\text{MMG}} \leftarrow P_{m^2} > P_{m^1}$
10:             $\text{score}_{m^1,m^2} \leftarrow \text{MCC}(y_{\text{MMG}}, \begin{bmatrix} \text{zeros}(N) \\ \text{ones}(N) \end{bmatrix})$
11:             if $\text{score}_{m^1,m^2} > Max$ then
12:                 $Max \leftarrow \text{score}_{m^1,m^2}$
13:                 $M_i^1 \leftarrow m^1$
14:                 $M_i^2 \leftarrow m^2$
15:             end if
16:         end for
17:     end for
18: end for
19: return $\{M_1^1, ..., M_k^1\}$ & $\{M_1^2, ..., M_k^2\}$
```

representatives, and the class with the highest likelihood is explained as the class of that sample.

Figure 4.4 shows the MCC results vs. the number of representatives, k, for the training and test sets. Note that MCC values are generally less than accuracy values. The results show TRUST is highly successful in explaining the primary AI's output on unseen data with an average of 90.89% MCC or 98% accuracy.

We have analyzed the performance in terms of time consumption, using the NSL-KDD dataset as an example. These experiments were run on a standard laptop with an Intel Core i7 CPU, 16 GB RAM, and Windows OS (no GPU). The steps to build the TRUST model only need to be done once, and after that, the model is reusable to easily explain the labels of future observations. Figure 4.5a shows the computation time to build the TRUST model (including

Algorithm 6 Fast Grid Search with $C = 2$

```
Function: modeSelectFastC2
Input: Representatives of class "1": $\{R_1^1, ..., R_k^1\} \in \mathbb{R}^{N_1 \times k}$;
    Representatives of class "2": $\{R_1^2, ..., R_k^2\} \in \mathbb{R}^{N_2 \times k}$;
    Zone $Z \in \mathbb{R}^{2 \times 2}$
 1: for each $(R_i^1, R_i^2)$ do
 2:     Divide zone $Z$ into $z_1, ..., z_d$
 3:     $Max \leftarrow 0$
 4:     for each $z_j$ do
 5:         $(m^1, m^2) \leftarrow z_j$'s center
 6:         $\mathrm{MMG}_i^1 \leftarrow$ fit $R_i^1$ to a guassian with $m^1$ modes
 7:         $\mathrm{MMG}_i^2 \leftarrow$ fit $R_i^2$ to a guassian with $m^2$ modes
 8:         $P_{m^1} \leftarrow p_1(R_i|\mathrm{MMG}_i^1)$                    ▷ $R_i = \begin{bmatrix} R_i^1 \\ R_i^2 \end{bmatrix}$
 9:         $P_{m^2} \leftarrow p_2(R_i|\mathrm{MMG}_i^2)$
10:         $y_{\mathrm{MMG}} \leftarrow P_{m^2} > P_{m^1}$
11:         $\mathrm{score}_{m^1,m^2} \leftarrow \mathrm{MCC}(y_{\mathrm{MMG}}, \begin{bmatrix} \mathrm{zeros}(N) \\ \mathrm{ones}(N) \end{bmatrix})$
12:         if $\mathrm{score}_{m^1,m^2} > Max$ then
13:             $Max \leftarrow \mathrm{score}_{m^1,m^2}$
14:             $Z_i \leftarrow z_j$
15:         end if
16:     end for
17:     modeSelectC2($\{R_i^1\}, \{R_i^2\}, Z_i$)
18: end for
19: return $\{M_1^1, ..., M_k^1\}$ & $\{M_1^2, ..., M_k^2\}$
```

the factor analysis, picking the representatives, and estimating their density functions) versus the number of representatives, k, was chosen to build the TRUST model. The time to search for the number of modes is not included in this graph. As expected, by increasing k, it takes more time to build the model since we have more representatives that they require to calculate their density functions.

Algorithm 3 and Algorithm 4 to determine the number of modes for a representative are compared in Figure 4.5b. The time consumption is calculated versus the number of samples. The search zone is a 20 by 20 grid, and when using Algorithm 4, it was divided into sixteen 5×5 sub-zones. As seen in this figure, Algorithm 4 speeds up the mode search by more than ten times.

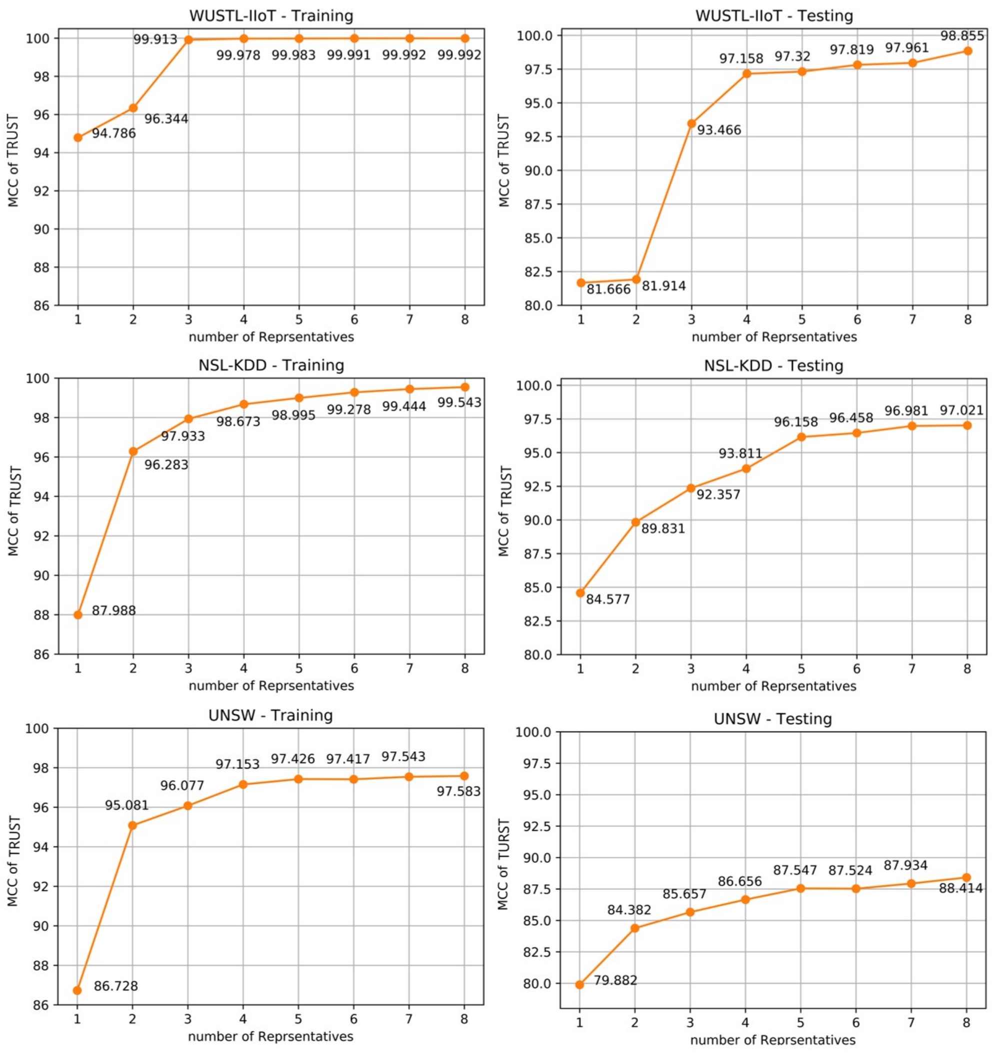


Figure 4.4: MCC scores of TRUST XAI on the training sets and the test sets

Next, the time for reasoning the labels of test samples by the TRUST model is examined. This step is done after the core of TRUST is built and is ready to label the unseen data as the class with the highest likelihood. First, TRUST labeling time consumption as a function of different numbers of representatives, k, is plotted in Figure 4.5c. As seen in this figure, the compute time is pretty insignificant. Second, Figure 4.5d shows the time consumption as a function of the number of samples.

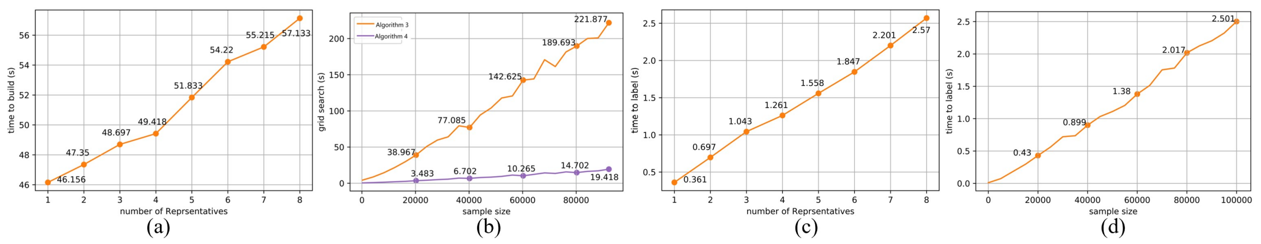


Figure 4.5: TRUST's Time consumption of (a) building the core model based on the number of representatives, k, (b) comparing Algorithm 3 and Algorithm 4, (c) labeling 100,778 samples vs. the number of representatives, k, (d) labeling time vs. the number of samples, N. In the last two figures, the training set is used because it has a larger number of samples

As mentioned before, our model has a significant improvement over the XAI models that require repeating the whole process for every single sample, such as LIME. The repetition makes their model time-consuming and inefficient when dealing with a large number of samples. On the contrary, our technique consists of a core model which is built only once. Once it is built, it is ready to explain the labels of future unseen data. It is fast and efficient when dealing with a significant amount of unseen incoming data. Figure 4.6a demonstrates this claim. Note that the TRUST's time of 212.65 s (3.5 minutes) includes the time to build the model from the 100,778 training samples. The time includes factor analysis, picking eight representatives, running the mode grid search (Algorithm 4) for them, estimating their density functions, and labeling the 8000 test samples. Our model is almost 25 times faster than LIME, which makes a significant difference in real-time applications. Next, we have compared the performance of our model with LIME using the accuracy metric, Eq. 2.1. Here, we used only five representatives in our TRUST model and calculated the accuracy of the test labels. As seen from Figure 4.6b, our model has performed better even with only five representatives, except for UNSW, where LIME has slightly higher accuracy (94.86% vs. 93.76%). By increasing the number of representatives, our model would be significantly superior to LIME in all cases. In terms of the difference in explainability methods between TRUST and LIME, we argue that our model produces more straightforward explanations

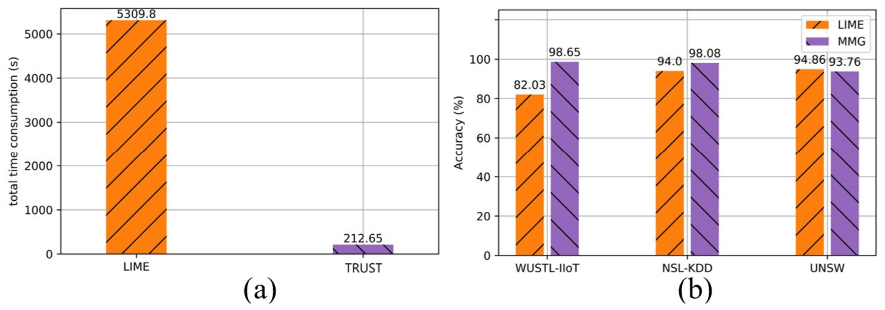


Figure 4.6: Explaining 8000 test samples; (a) time consumption of our TRUST model vs. LIME, (the TRUST's built time is also included), (b) performance of TRUST vs. LIME

Table 4.7: Log-likelihood

(a) Log-likelihood of belonging to Attack class

	1st Rep.	2nd Rep.	3rd Rep.	4th Rep.
Attack samples	5.51	5.46	4.40	4.27
	1.56	-2.46	-1.19	1.54
	-1.79	-2.06	0.68	-3.06
Normal samples	-3.61	-2.09	-2.30	-5.21
	-6.99	-2.16	-3.12	-2.99
	-3.56	-2.19	-10.09	-4.72

(b) Log-likelihood of belonging to Normal class

	1st Rep.	2nd Rep.	3rd Rep.	4th Rep.
Attack samples	-0.30	1.87	-0.04	-0.86
	0.08	-4.92	-1.11	-0.89
	-2.52	-2.16	-0.31	-6.18
Normal samples	-3.68	1.86	-2.11	-5.16
	-3.25	1.83	-2.76	-0.94
	-3.02	1.75	-2.41	-0.90

(c) Comparison of log-likelihood

1st Rep.	2nd Rep.	3rd Rep.	4th Rep.
5.51 > -0.30	5.46 > 1.87	4.40 > -0.04	4.27 > -0.86
1.56 > 0.08	-2.46 > -4.92	-1.19 < -1.11	1.54 > -0.89
-1.79 > -2.52	-2.06 > -2.16	0.68 > -0.31	-3.06 > -6.18
-3.61 > -3.68	-2.09 < 1.86	-2.30 < -2.11	-5.21 < -5.16
-6.99 < -3.25	-2.16 < 1.83	-3.12 < -2.76	-2.99 < -0.94
-3.56 < -3.02	-2.19 < 1.75	-10.09 < -2.41	-4.72 < -0.90

(d) Total log-likelihood

Total
19.64> 0.67
-0.55>-6.84
-6.23>-11.17
-13.21<-9.09
-15.26<-5.12
-20.56<-4.58

which are easier to understand. LIME is content with local approximations of the primary AI with other simple AI or ML models as a way of explainability and interpretability. Whereas, we model the behavior of the primary AI's output with statistical measures. As statistics are more commonly a part of one's educational background, they are much easier to understand compared to grasping the ideas behind the internal rationale of an AI or ML model.

4.5 How TRUST Explains The Primary AI Results to The User

This section discusses how the TRUST explainer can explain the labels to a human user simply and interpretably on the NSL-KDD dataset as an example. As mentioned before, it is safe to assume that the user of the AI-based security system has a basic knowledge of AI, mathematics, and statistics. Each traffic packet in the NSL-KDD dataset is a 40-dimensional vector, but the TRUST explainer would calculate the representatives and reduce each sample to a four-dimensional vector. This process of dimension reduction is simply explained as follows.

The TRUST explainer modifies the feature values to remove redundancy in the data for a more transparent statistical analysis. By factorizing and removing their interdependencies and projecting them onto orthogonal dimensions, TRUST removes any correlation in the data. The transformed feature values are the factors introduced before. Then, utilizing a correlation metric (here, mutual information), the most revealing factors determining the class labels are chosen as representatives. Afterward, TRUST models the statistical behavior of the representatives and estimates their density functions with unique multi-modal distributions.

As mentioned before, statistics are more easily understood than the AI or ML models used as the XAI. For elaboration, we have shown an example in Figure 4.7. It is trivial for the user to understand the likelihood differences of a new sample belonging to the pdfs of each class's representative. In this example, the likelihood of belonging to the attack class is higher.

To show the explanation process of TRUST, we randomly chose six samples (three attacks, three normals) from the test set. The log-likelihoods of belonging to each class using individual representatives based on Eq. 4.6 are computed. The results of each instance belonging to the

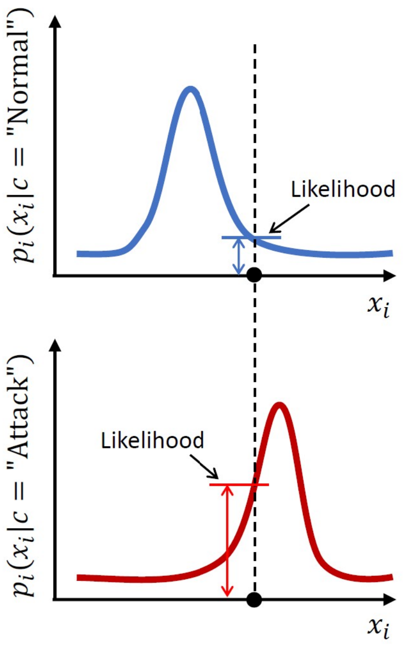


Figure 4.7: An example of pdf curves for the ith representative

attack class and belonging to the normal class are shown in Tables 4.7a and 4.7b, respectively. After an element-wise comparison of the values in these two matrices, the class that has a higher likelihood is marked in Table 4.7c. Notice that not all the representatives give us the right class all the time. For example, the 1^{st} representative makes mistakes with the 4^{th} sample. Similarly, the 3^{rd} representative makes mistakes with the 2^{nd} sample. However, when all four representatives are used to calculate the overall log-likelihood, as shown in Table 4.7d, the labeled class is correctly explained for all six instances.

4.6 Conclusion and Future Direction

XAI has attracted extensive attention recent years and has been treated as a necessary trend toward the next generation of AI. However, there is a significant lack of research work in this domain, specifically for numerical applications (compared to image applications). Meanwhile, several critical applications such as cybersecurity and IoT are highly dependent on numerical data. The available work in the literature falls short from several aspects, such as trade-offs among performance, speed, reusability, and complexity, as discussed in this chapter. Our proposed TRUST XAI is a breakthrough in this domain in integrating transparency, being independent, and comparability (in a statistical aspect) into AI-based systems. Here, we take the lead to use statistical principles to build XAI models that can accurately estimate the distributions of the AI's outputs and calculate the likelihood of new samples belonging to each class. Moreover, we have proven the superiority of our model to a currently popular XAI model, LIME. Our results show that our model is 25 times faster and more accurate than LIME making our model a great candidate for real-time and critical applications.

Despite TRUST's superior performance, it has some limitations as well. Due to using information gain in picking the representatives, TRUST might overfit to the training set. This would lead to poor performance of unseen data. On the other hand, if the Gaussian assumption cannot be made or the probability distribution of data changes, the output of TRUST would not be reliable. Also, the assumption of samples being drawn independently is very important.

As future direction, the selection of representatives, we can take into account other metrics such as gini index. In addition, we can extend the model to consider other possible probability distributions (e.g., Poisson). Also, removing the independence assumption requires approximating the joint probabilities between each pair of features.

Chapter 5

High Computational Load of AI

When dealing with the IoT technology, especially industrial IoT, two manifest challenges leap to mind. First is the massive amount of data streaming to and from IoT devices, and second is the fast pace at which these systems must operate. Distributed computing in the form of edge/cloud structure is a popular technique to overcome these two challenges. In this chapter, we propose ADDAI (Anomaly Detection using Distributed AI) that can easily span out geographically to cover a large number of IoT sources. Due to its distributed nature, it guarantees critical IIoT requirements such as high speed, robustness against a single point of failure, low communication overhead, and scalability. Through empirical proof, we show the communication cost is minimized, and the performance improves significantly without the need of sending the raw data. ADDAI provides predictions for new random samples with an average success rate of 98.4% while reducing the communication overhead by half compared with the traditional technique of offloading all the raw sensor data to the cloud.

5.1 Introduction and Motivation

As mentioned previously, IoT integrated into the ICSs is called Industrial Internet of Things or IIoT. IIoT systems are the foundations of the critical infrastructures of a nation. Industrial processes such as smart manufacturing, oil and gas exploration, water distribution and treatment, etc., rely heavily on these systems. Utilizing IoT technology in ICSs enhances network intelligence in the optimization and automation of industrial operations.

Moreover, the impact of AI analysis on the rapid growth of the IoT is undeniable. IoT has become a fundamental part of both personal lives and the industrial environments. The integration of AI and IoT automates connections, interactions, and data exchange among machines and devices for comprehensive functionality and higher efficiency without requiring any human in the loop. This automation needs high-speed computations; however, the heavy computational load of AI-based algorithms leaves a burden on the system's data processing speed. This concern is even more troublesome in systems such as IIoT, where real-time operations are essential.

According to the annual report of Stanford University's AI Index, the speed of AI is outpacing Moore's Law. Since 2012, AI's required computing power has increased 300000 times compared to the expected seven times increase under Moore's law [83]. The computation power is expected to double every two years based on Moore's law, while the amount of computing power required for AI doubles every 3.4 months . This proves the unsustainable computing power of AI and the fact that in several years, reaching a general intelligence will not be through a highly powerful complex AI algorithm, but rather through collective intelligence. Collective intelligence means distributing the computation and making decisions through several learning components concurrently and in parallel. This fact emphasizes why distributing the computations in an AI-based IoT system is essential.

Requiring all the raw data to be offloaded to one centric computing element for processing would cause a bottleneck in the system. In heavy computational scenarios, the created congestion will affect the real-time operations and slow down the critical functions.

On the other hand, traditional centralized computing is not capable of carrying out, keeping up, and adapting to the required security operations. At the same time, all the sensor data is offloaded into a large and complex AI. It also suffers from an important security vulnerability, i.e., a single point of failure, which negatively impacts the main goal of IIoT systems, high availability and reliability. Also, it comes with high latency and communication costs. Therefore, our goal in this project is to reduce the need to send all the field data to a central remote computing processor. This will lead to a promising architectural solution to a fundamental network problem, latency.

Despite the increased interest in using rapid computations in AI analysis, most research works focus on distributing the training process, usually through updating model parameters. To the best of our knowledge, there has been no research that practically fragments the process of anomaly detection using AI models in a granular fashion that would significantly decrease the computation overhead. Not only the training, but also labeling new data should be distributed in a way that their integrity is not compromised. Our proposed model fragments the decision making-process of AI-based anomaly detection models in a granular fashion. It can also make decisions locally and later take advantage of those decisions when it comes to classification tasks in the upper layer hierarchy.

To evaluate and provide a practical instance for ADDAI, we use our datasets: “WUSTL-IIoT-2021," a dataset generated from our prior work on our industrial IoT (IIoT) testbed [107]. Following the previous chapter, we will refer to our dataset as “WUSTL-IIoT" in this chapter as well. We empirically prove that our proposed DAI model predicts the labels of unseen data with a high success rate.

5.2 Related Work

DAI is an immense area that includes the distribution of different parts of AI models, such as the datasets, training, tasks, etc. Distribution among different computing entities can be decided based on their differences in energy consumption, memory restrictions, computing power, and many more factors. Additionally, a centralized solution is not even an option when data is inherently distributed or too big to store on a single device. There exists several comprehensive surveys related to this area such as [68], [71], [3], [84], and [99].

On the other hand, there are situations in which it is beneficial or even required to isolate some subsets of the data. These concerns usually happen when privacy issues are involved. In [1], a framework of differential privacy is considered, and a deep neural network with a modest total privacy loss is developed. Another technique to training models in a privacy-sensitive context is utilizing distributed ensemble models. This guarantees the complete separation of the training data subsets where privacy needs to be preserved. [87] proposes a federated learning approach that uses ensemble distillation in a medical relation dataset. The suggested technique also overcomes the communication bottleneck caused by the need to upload a large number of parameters in regular federated learning models. Another challenge in these kind of models is that a method needs to be found that properly balances each model's input for an unbiased training result.

Another popular application of DAI is in unreliable networks, where we have a lossy network and cannot really get a guarantee that sent information will successfully get to the supposed receiver. In [102], such scenarios are studied. To overcome this challenge, parameter servers are introduced to store the parameters of the learning model (e.g., the weights of a neural network). These servers serve the parameters to local units (Workers) in charge of processing the data and computing updates to the parameters. Each server connects to all the local units and serves a partition of the model, and each local unit holds a replica of the whole model. However, every communication link between each server and each local unit has a non-zero probability of being dropped.

DAI is a well-known approach in edge-cloud computing. The concept is to move all or some of the training workloads from the central computing unit (e.g., the cloud) to the edges. This way, we reduce the enormous network communication overhead and provide low-latency solutions. To name some challenges in this approach, we can mention parallel training, model synchronization, and workload balancing to address the imbalance of workload and computational power of different edge nodes. For instance, [16] studies a case study of utilizing DAI in video surveillance systems. Popular services are also cached on the edge servers, so that the latency can be reduced even more, and the computation can be offloaded easily. However, only the cost of bringing services to the edge is considered, and the cost of transmitting data between the edge and cloud server is assumed negligible.

Not all the elements in the IIoT systems have the same computing capabilities. The distribution of the tasks among the elements of the hierarchy must be fair. Therefore, their different computation and communication capabilities should be assessed. The trade-off between cost and capacity of the resources is another important consideration for optimizing resource sharing. There is an extensive study and survey on resource management of different levels of hierarchy and their constraints in [91] and [56].

Another approach in the distributed AI is to divide the tasks into micro-services and spread them out in a distributed fashion. These tasks can be data filtering and cleansing, training, testing, labeling, security computation, etc. Different stages of learning development can also be divided into layers among a few high-capacity processing entities. Afterward, the developed model can be accessed simultaneously by several smaller processors to predict the labels of new instances, which is a low computation task. The same mechanism can be applied to other computation tasks. As another example, breaking the training set or the training process into several parallel local tasks can also provide an early result at the local layer and help deal with massive datasets and develop scalable learning. Some examples of research works in this concept include [46], [98], and [45].

5.3 Background

ADDAI is composed of two learning models, autoencoders and AdaBoost. For the sake of completeness, we first provide the basics of how these two models function before diving into the details of our proposed model.

5.3.1 Autoencoder

Autoencoders can work with voluminous unlabeled data and extract useful features with good accuracy. This is specifically useful in industrial systems since a large percentage of data is unlabeled, and it is very time consuming and labor intensive to be tagging this large amount of data. Autoencoders can be used to extract useful features from high dimensional sensor data.

Autoencoder is a type of neural network with a bottleneck layer in its network, forcing it to produce a compressed knowledge representation of the original input (also known as the code). The network consists of two parts, an encoder part that produces the code and a decoder that reconstructs the input from the code. By learning a smaller size code, autoencoders can

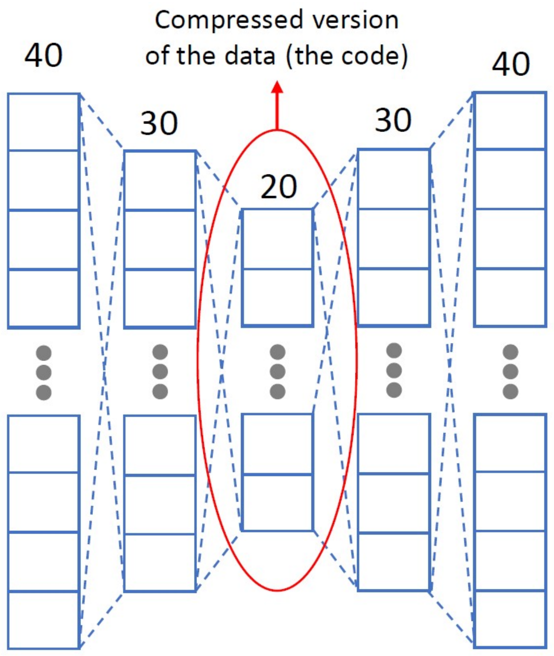


Figure 5.1: Diagram of an autoencoder. The middle layer, which is called the code, is the compressed version of the input

work with voluminous unlabeled data, such as sensor data in industrial systems. Learning a smaller size code forces the autoencoder to capture the most salient features of the input and extract useful features with good accuracy [30].

Autoencoders provide dimension reduction and the output data is latent variables. They remove the redundancy in the data and combine all the features together rather than just simply pick the important features from the inputs, which is the most common way of dimension reduction.

Suppose we have a training set $X \neq \emptyset \in \mathbb{R}^{N \times K}$, where each sample x_i in $X = \{x_1, x_2, ..., x_N\}$ is a $1 \times K$ vector (i.e., K features). An example is shown in Fig. 5.1. In this example, we have a dataset in which each sample has $K = 40$ features, and the size of the produced code is 20.

Any neural network can be uniquely identified and built by its hyper-parameters. There are four hyper-parameters for an autoencoder which are weights $\boldsymbol{W}$, biases $\boldsymbol{b}$, code size, and the number of hidden layers (when the number of nodes per layer and the loss function are fixed). The weights specify the weighted connections of neurons between every two adjacent layers, while the biases shift the activation function by adding a constant.

Choosing the size of the code and the number of hidden layers both have specific trade-offs. The size of the code should not be too small. First, we might lose more information from the data. Second, it would be tough to capture all the relationships among the input features. This causes under-fitting. Meanwhile, the size of the code should not be too large. The network might simply overfit the code layer by learning the noise and small details of input data.

On the other hand, the number of hidden layers should not be too high. The gradient effects of the first layers become too small, and the input becomes almost irrelevant to the code. On the contrary, if the number of hidden layers is too small, it cannot extract useful relations. In neural networks, the first layers usually extract very simple relations like lines and curves, but deeper layers extract more complex features.

Construction error is another important concept in autoencoders. As mentioned before, the decoder part tries to mimic the input layer in the output layer. The more similar these two layers are, the more accurately the autoencoder extracts the code. The most popular way of optimizing the autoencoder's parameters to minimize the construction error is using the mean squared error (MSE), Eq. 5.1

$$L(x_i, x_i') = 1/K \sum_{j=1}^{j=K} (x_{i,j} - x_{i,j}')^2 \tag{5.1}$$

where, x_i, is a sample for the input layer (e.g., the leftmost 40 dimensional layer in Fig. 5.1) and $x_i{}'$ is the output layer (e.g., the rightmost 40 dimensional layer in Fig. 5.1).

5.3.2 AdaBoost

AdaBoost, which is short for Adaptive Boosting, is a popular boosting technique. Boosting is an ensemble method that improves the performance of a number of weak learners and builds a strong learner. Decision trees are the most suited and commonly used weak learners in AdaBoost. In this technique, the weak learners are added sequentially and trained on repeatedly modified versions of the data (i.e., different weights for each instance). Training stops when a pre-defined number of weak learners are trained, or no further improvement can be made. Afterward, the predictions from all the weak learners are combined through a weighted sum or vote to be decided as the final decision, Eq. 5.2. This information infusion is the basis of ensemble learning. Each learner has a stage value that is used to weight its predictions in the final decision on the instance' label. The stage value is simply based on their performance so that the more accurate the model generally is, the more contribution to the final prediction it has.

$$L_T(X) = \sum_{t=1}^{T} \alpha_t l_t(X) \tag{5.2}$$

where $\alpha_t = \frac{1}{2}\ln(\frac{1-\epsilon_t}{\epsilon_t})$ is the weight of t'th weak learner, in which $\epsilon_t = (\sum_{y_i \neq l_t(x_i)} w_{i,t})/(\sum_{i=1}^{N} w_{i,t})$ is its weighted error rate.

In AdaBoost, we initially give each instance in the training dataset equal weights. As the training progresses, at each step, it gives more weights to individual instances that were classified wrong and less to those already correctly classified. The updating rule for the weight of each training sample is based on Eq. 5.3.

$$w_{i,t+1} = w_{i,t} \times e^{-y_i l_t(x_i)\alpha_t} \tag{5.3}$$

If we assume $y_i \in -1, 1$, when the prediction of the weak learner is correct (i.e., both y_i and $l_t(x_i)$ are -1 or 1), the coefficient to updating the weight is $e^{-\alpha_t}$. This means we decrease the weight. When they are different, the updating term will be e^{α_t} which means more weight is assigned to a misclassified sample. [6]

5.4 Proposed ADDAI Model

We propose a layer-wise prediction scheme. These layers include local processor units, the edge, and the cloud. The local processor is in direct contact with a group of sensors working together. We assume near zero communication delay between the sensors and the local processor. The edge can be a middle point between the local processors and the cloud for lower latency computations than cloud offloading. For simplicity, here, we consider a two-layer hierarchy, i.e., the local-cloud scenario. Our proposed method can be easily generalized to a general local-edge-cloud scenario. In the following, we discuss the formulate the problem and details of the proposed technique.

5.4.1 Problem Formalization

In our proposed scenario, the tasks are distributed among these components (i.e., local and cloud) for higher efficiency in terms of latency and communication overhead. One important task that is done locally and distributed all over the network is anomaly detection. This way, the anomalies can be detected by the local processing units quickly without the need for

[6]In our dataset, class normal is labeled as 0, but for the sake of weight updates in AdaBoost, we represent the class normal with -1 instead of 0. This is just a label and does not have any numerical value.

sending the data to any other local unit or the cloud. On the other hand, the computation-intensive tasks are done in the cloud. These tasks include training the local models and further investigating the label of the data when we are still uncertain about it despite the local processing.

Two scenarios can be defined whether we can trust the classification of the local model at the local processing unit or not? If yes, local decisions are made, and then only the final results are sent to the higher hierarchy (e.g., the edge and/or the cloud). If the classifications of the local model cannot be trusted, e.g., due to low accuracy rates, further investigation is required on the sample's label. We design the local units such that at least a compressed version of the data that is much smaller compared to the original raw sensor data that would be sent to the cloud. With this scheme of layer-wise computations, the communication between the local processors and the cloud will be decreased to a significantly lower rate.

It is important to note that, we suppose that the network data from the local units have approximately the same topology and format. Hence, at a given point in time across all the local units, network readings are similar if no anomaly occurs. If this assumption cannot be made and the system is a large-scale network with different protocols and data, the proposed model can easily scale out and group the similar nodes in the network into a cluster. Each cluster would require a tailored autoencoder and the cloud would maintain all the autoencoders separately.

5.4.2 Architecture

We develop an autoencoder as the proper learning framework for local processing units. Fig. 5.2 shows the diagram of the proposed model. Since autoencoders are also known as dimension reduction technique, only a more abstract and less redundant information is sent up to the cloud. This helps to reduce the required network communication significantly when it is necessary to send the data to the cloud. With their produced compressed version of the input, the communication cost significantly drops. This design also prevents the cloud from seeing the actual data because latent variables composed from the data features are sent to the cloud instead of the raw data from sensors.

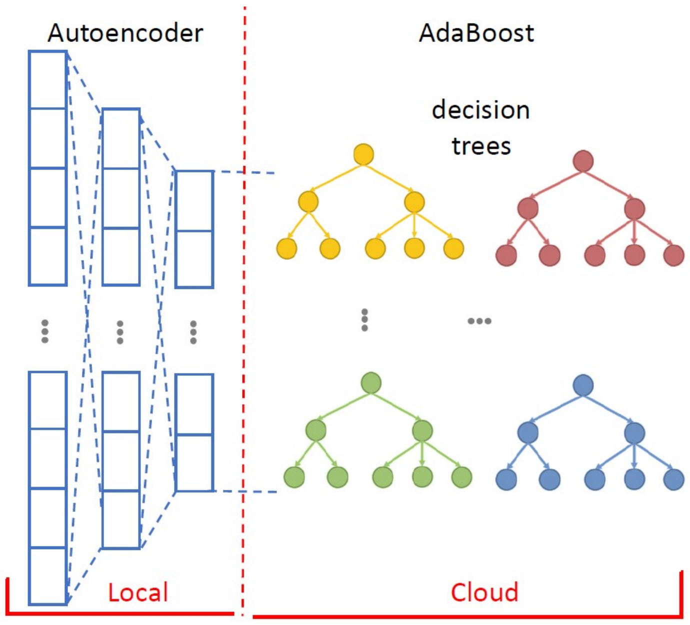


Figure 5.2: Diagram of the proposed model. The raw data first is fed into a local autoencoder to compress and resize; then, they will be fed into an AdaBoost model in the cloud to decide on their label

We train the local autoencoder by normal data only. The achieved model parameter ($\boldsymbol{W}$ and $\boldsymbol{b}$) in the trained autoencoder are used to accurately produce the compressed data (representing code) of the normal samples. Therefore, the compressed data is a model for the normal data. When an attack sample is fed to the model, the reconstruction error is very large, which is how the anomalies are detected. The larger the error is, the more confident we are that the sample is an attack.

Since the training might be computationally heavy for some of the local units, the cloud trains the autoencoder model using the data from all the sensors, $X = \{x_1, x_2, ..., x_N\}$. Here, N is the total number of samples from all the sensors across the network. After the training is done, the cloud sends the updated model parameters $\boldsymbol{W}$ and $\boldsymbol{b}$ back to the local units to build a local version of the autoencoder. Please bear in mind that training is a one-time process, and we can assume the features of the data does not change drastically over time, so the trained model is generalized enough and stays good for a long period of time.

If the predictions of the local processing units are trustworthy, in case an anomaly is detected, they can quickly take action by alarming the control center. However, sometimes the compressed data is sent to the cloud for further investigation. In the cloud, we utilize three different trained AdaBoost models. One of these models is more sensitive on detecting the normal class; another one is sensitive to the attack class, and the third one is a neutral model that pays fair attention to both classes. In other words, these three models are trained in a way that they would have different sensitivity/recall scores for different classes of the data. Since AdaBoost works with weights for individual instances, we have assigned class weights at the decision tree levels (through the weak learners). This class weights can also be very helpful when dealing with imbalanced datasets, where the minority class is usually overlooked by the model. The loss function for each of the AdaBoost model can be modeled as in Eq. 5.4.

$$\text{loss} = \frac{1}{N}\sum_{i=1}^{N} cw_0(y_i \times \log y_i') + cw_1((1 - y_i) \times \log(1 - y_i')) \tag{5.4}$$

where cw_0 and cw_1 are the class weights in the case of binary classification. A grid search is conducted to get the optimal value of these weights.

In our proposed model, despite the fact we might not fully trust the local decisions on the data labels, we want to take advantage of them. We utilize them in a new concept we call *local ranges* to identify the boundaries for which samples would be fed to which AdaBoost model in the cloud.

5.4.3 Local Range

The local range is a critical component of our proposed architecture. It determines the three ranges of the samples fed to each AdaBoost model in the cloud. It is specific for each local processing unit and depends on how certain we are in the predictions of the samples made by that local unit classifier. A local unit has an accuracy score between 0 and 1; we call it f_i for i'th local unit. Therefore, the ratio of the number of samples that are misclassified by the autoencoder in the i'th local unit is $r_i = 1 - f_i$.

On the other hand, when the local unit calculates the reconstruction errors, we can sort them in ascending order. Let us call the sorted array Err_i. Local autoencoder would classify any sample with a reconstruction error below a threshold, η_i, as normal and any sample above that as an attack. With good approximate, we can assume that misclassification happens around the threshold. We want to feed these samples to the regular model as it is partial to both classes. Therefore, we define the range of the reconstruction error of the samples that should be fed to the regular model is defined as

$$\{\eta_i - Err_i[idx - \lfloor(r_i \times N_i/2)\rfloor], \eta_i + Err_i[idx + \lfloor(r_i \times N_i/2)\rfloor]\} \tag{5.5}$$

where η_i is the threshold calculated by the autoencoder for the i'th local unit based on its pick performance on the training data specific to that local unit. idx is the sample index whose reconstruction error is the closest to the value of η_i, and N_i is the number of training samples in the local unit i.

As shown in Fig. 5.3, we make the decision on which sample to be fed to which model based on the local range and the value of the reconstruction error of the sample. The samples with reconstruction errors within the local range to be fed to the regular model, and below or above that to the normal and attack models, respectively. This figure shows an example of reconstruction error for a set of test samples. The more uncertain we are about the local decisions, the wider the boundary for the regular model is.

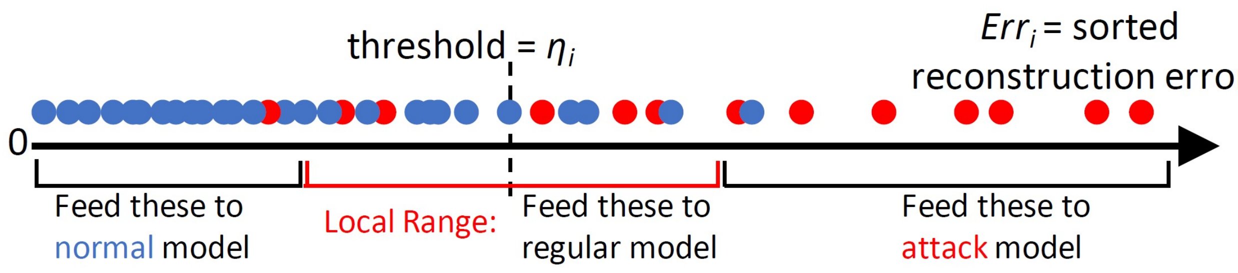

threshold = η_i
Err_i = sorted reconstruction error
0
Feed these to normal model
Local Range:
Feed these to regular model
Feed these to attack model

Figure 5.3: Local decisions based on the reconstruction errors from the autoencoder

5.4.4 Communication Cost

Note that the intermediate output is designed to be much smaller than the sensor input (e.g., a raw image from a video camera), and therefore drastically reduce the network communication required between the end device and the cloud. The communication cost for offloading the data to the cloud, k, can be formulated as follows:

$$k = 1 \text{ (for } \mathcal{C}) + 4 \text{ (for reconstruction error)} + 4 \times h \tag{5.6}$$

In this equation, $\mathcal{C}$ is the predicted class by the local unit, and one byte is assigned for it. 4 bytes are dedicated for the reconstruction error. h is the length of the code layer of the autoencoder. 4 bytes are used to represent a signed single-precision floating-point number.

5.5 Case Study: Distributed AI for IIoT Security

Sophisticated high computational AI models are vastly utilized in industrial settings. Intrusion detection systems (IDS) for network security is one of the common applications in which AI is a popular approach . Utilizing ADDAI would distribute and speed up the detection process in these systems. As mentioned before, we use our lab scale IIoT system to run realistic and up-to-date experiments for the AI-based IDS.

5.5.1 The Learning Models

The dataset is the one collected from our testbed, which we refer to as “WUSTL-IIoT." The inputs to the learning models are the flow instances, as mentioned in the previous subsection. The output of the models can be a multi-class or binary classification. However, for simplicity, we have built ADDAI by treating the outputs of the model as binary classes with normal as 0 and attack as 1.

The autoencoder used in the local processing is programmed using the neural network module of PyTorch [66], containing the encode, decode, and forward methods. Depending on the code size, we use a different number of hidden layers. For instance, for a code size of 25, 7 layers are used in the model: input data layer, two encoding layers, code layer, two decoding layers, and the output layer, as shown in Table 5.1. After the first layer (i.e., the input layer), the local autoencoder has two hidden layers with 35 and 30 neurons. A hundred epochs with *adam* optimizer and a learning rate of 0.01 and the dropout regularization technique with 0.05 rate to avoid overfitting are utilized. For the activation function, we have used hyperbolic tangent, or tanh, function, which is defined as $\tanh(x) = (e^x - e^{-x}/(e^x + e^{-x})$. AdaBoost is used in the cloud with decision tree classifiers as the weak learner. The number of estimators in AdaBoost was set at 100. The scikit-learn library [67] was used to implement the AdaBoost models.

Table 5.1: Local autoencoder model hyperparameters

Parameter	**Typical value(s)**
# of layers	7
# of neurons per layer	40, 35, 30, 25, 30, 35, 40
# of epochs	100
Optimizer	*adam*
Dropout rate	0.05
Learning rate	0.01
Activation function	hyperbolic tanh function

5.5.2 Results

In this chapter, we evaluate the model's performance using three metrics, including Accuracy, MCC, and undetected rate (UR) shown in Eq. 2.1, Eq. 2.4, Eq. 2.3, respectively. In this experiment setup, we set the number of local units to be 3. We have randomly divided the WUSTL-IIoT dataset into three equal sized subsets. Fig 5.4 shows the exact number of samples of each class per local device.

Here, we test the performance of ADDAI assuming the worst-case scenario, where we cannot trust any of the local decisions on the samples' label, and they should be sent to the cloud for further investigation. However, we take advantage of the local analysis as mentioned before.

Our analysis starts by choosing the right code size based on the performance and the communication cost. For different values of h, $10, 15, 20, 25$, and 30, the associated communication

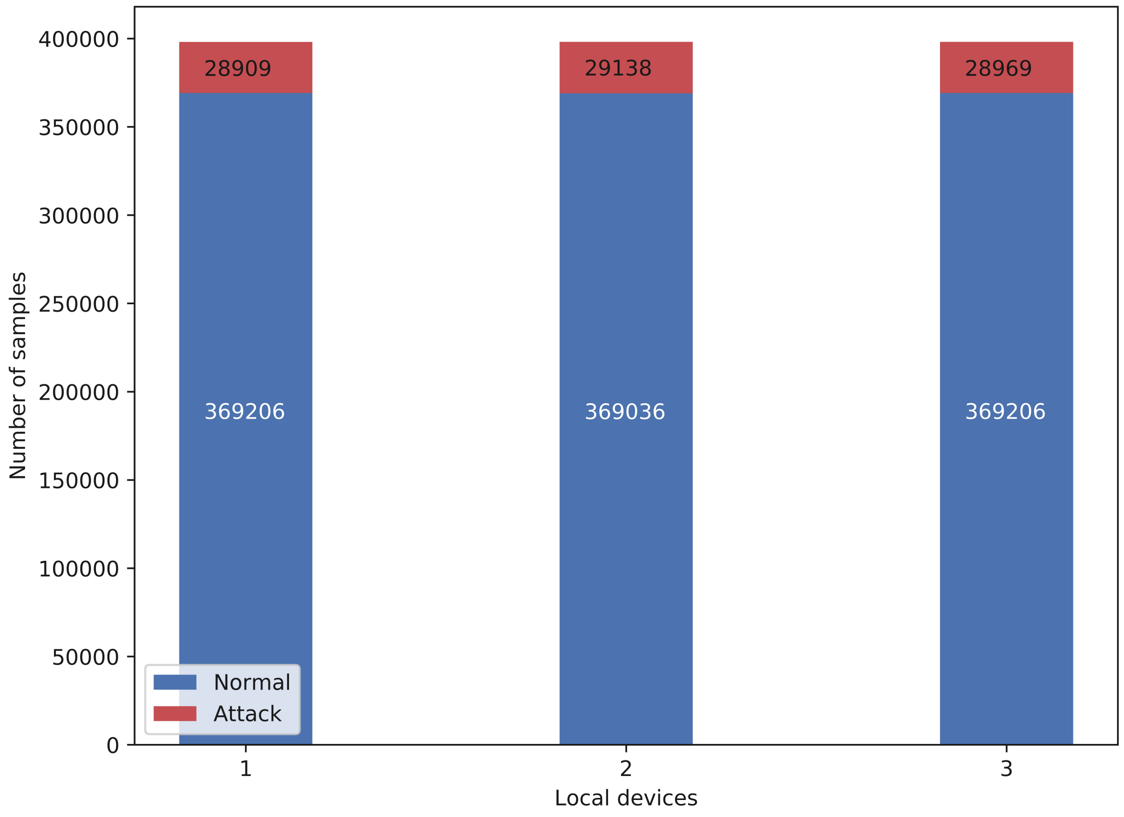


Figure 5.4: Number of samples from each class of data (Attack or Normal) at each local device

cost (Eq. 5.6) and the performance of the local devices are compared. Fig. 5.5 represents the change in the MCC results at the cloud for each of these values. The code size of 25 with an average of 105B communication cost produced the best result. Therefore, in the rest of the evaluations, we use the code size of 25 unless stated otherwise. For this setting, the stored autoencoder models on the local processing units require only 16 kB of memory.

Tables 5.2, 5.3, and 5.4 represent the performance results on the three local units. Each local unit splits its dataset into training and test sets with an 80:20 ratio. Since the autoencoder in the local units predicts the labels (regardless we can trust them or not), we have utilized them as the local performance on the data. The predicted labels in the local units are presented in (a) tables, while the cloud results after further investigation are shown in (b) tables.

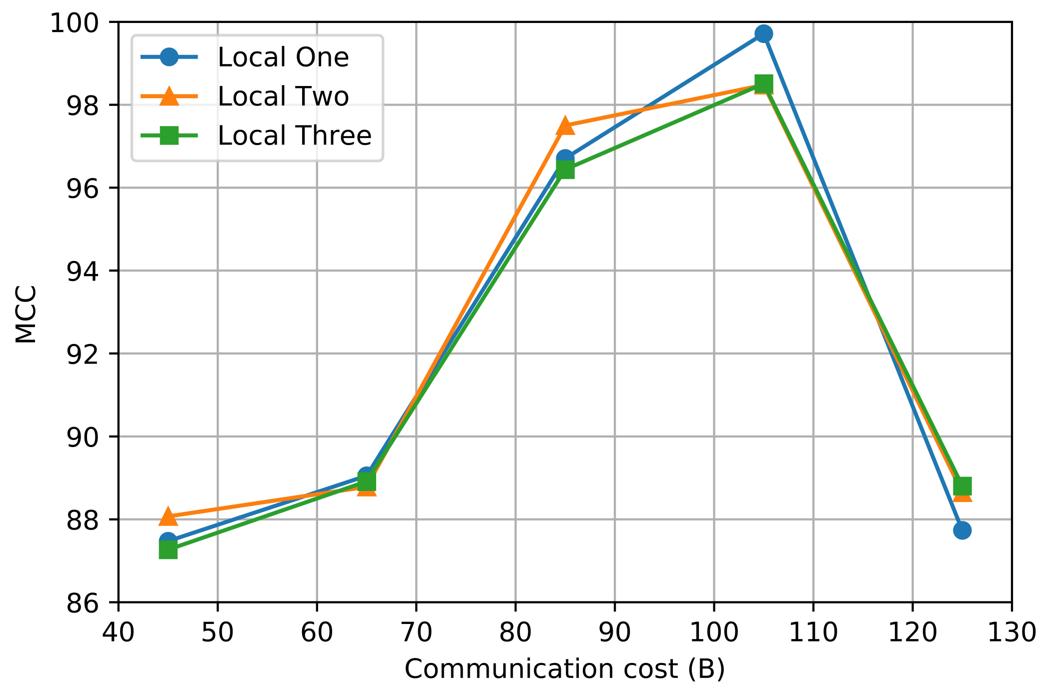


Figure 5.5: Communication cost vs. MCC scores for different code lengths

Table 5.2: Evaluation of the data from local device one

	Normal	Attack
Normal	73069	782
Attack	199	5573

(a) Local results

	Normal	Attack
Normal	73696	155
Attack	20	5752

(b) Cloud results

Table 5.3: Evaluation of the data from local device two

	Normal	Attack
Normal	73204	592
Attack	192	5647

(a) Local results

	Normal	Attack
Normal	73614	182
Attack	30	5809

(b) Cloud results

Table 5.4: Evaluation of the data from local device three

	Normal	Attack
Normal	73131	711
Attack	156	5637

(a) Local results

	Normal	Attack
Normal	73713	129
Attack	8	5785

(b) Cloud results

To get a concrete comparison of the performance improvement through our model, the performance metrics mentioned in Section 2.6 calculated from the values in the previous tables are summarized in Fig. 5.6 and Table 5.5. Please note that, for accuracy and MCC metrics, the higher the value, the better, while for UR, the lower the value, the better.

Each local unit splits its dataset into training and test sets with an 80:20 ratio. Since the autoencoder in the local units predicts the labels (regardless we can trust them or not), we have utilized them as the local performance on the data. The cloud results are derived from feeding different samples to different AdaBoost models (regular, normal, and attack) based on their calculated local ranges. Table 5.6 shows the utilized ranges for each local device. These numbers are calculated using Eq. 5.5.

Table 5.5: Summary of the local and cloud performances in all three datasets

	Accuracy	**MCC**	**UR**
One, local	98.76%	91.36%	3.44%
One, cloud	99.78%	98.39%	0.34%
Two, local	99.01%	93.03%	3.29%
Two, cloud	99.73%	98.07%	0.51%
Three, local	98.91%	92.38%	2.69%
Three, cloud	99.82%	98.74%	0.14%

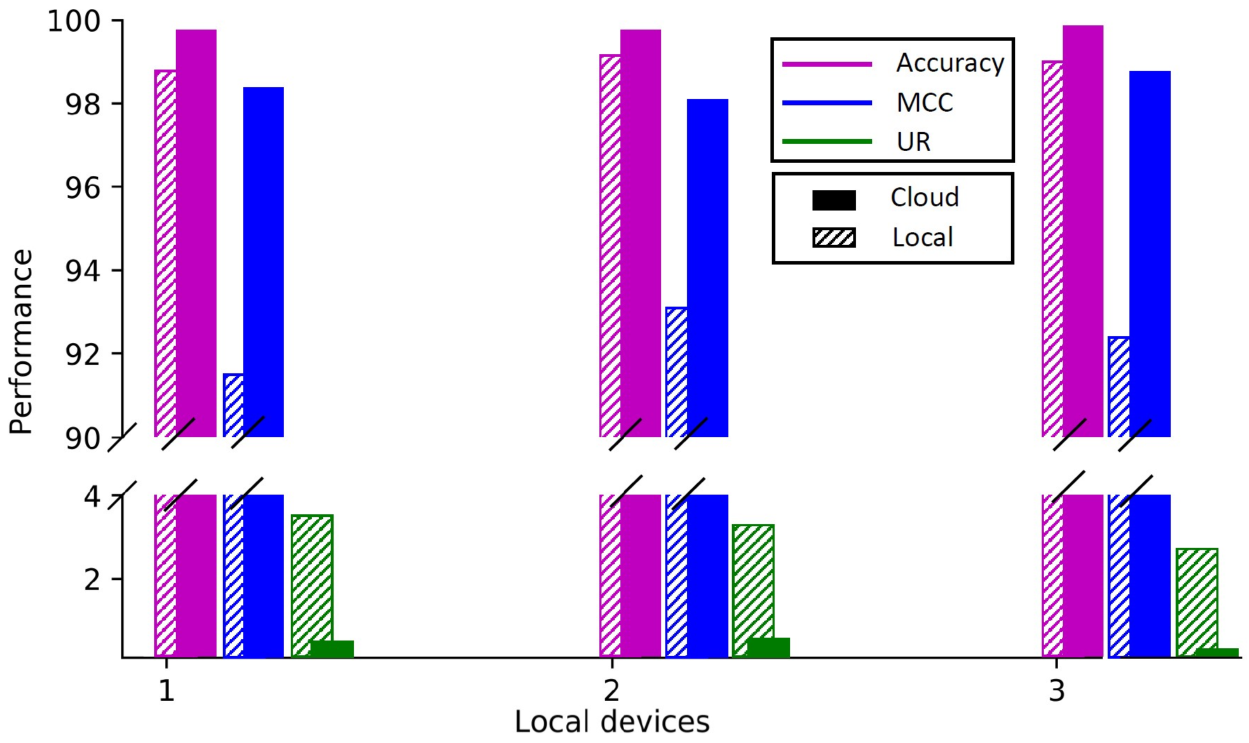


Figure 5.6: Different performance metrics comparison of the three local devices vs. their corresponding cloud results.

Table 5.6: Specifics of the tested dataset

Local device	$\boldsymbol{\eta_i}$	**Normal range**
One	0.43	[0.29, 1.38]
Two	0.58	[0.17, 1.43]
Three	0.42	[0.28, 1.35]

We also tested what would happen if we feed all the samples to only one of the AdaBoost models in the cloud. For simplicity, we combined the data from all the local devices. The results are shown in Fig. 5.7 and Table 5.7.

Table 5.7: The MCC results, if only one of the cloud models was utilized

	Normal	Attack
Normal	220213	1243
Attack	1198	16239
MCC = 92.45%		

(a) Regular model

	Normal	Attack
Normal	220293	1163
Attack	1250	16187
MCC = 92.51%		

(b) Normal model

	Normal	Attack
Normal	216932	4524
Attack	26	17411
MCC = 88.10%		

(c) Attack model

	Normal	Attack
Normal	220558	898
Attack	112	17325
MCC = 96.97%		

(d) ADDAI

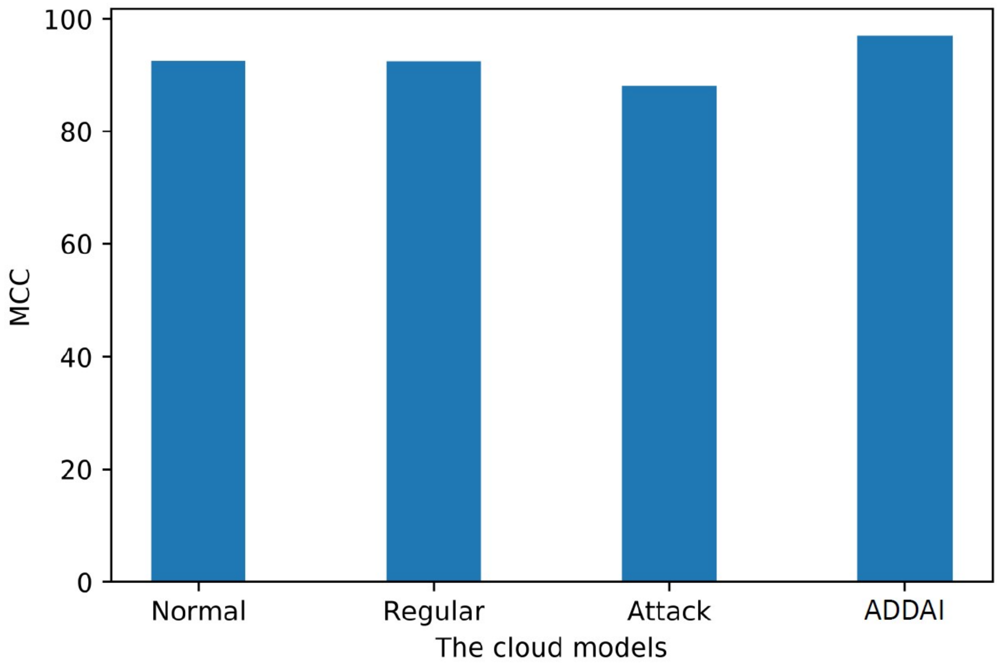


Figure 5.7: The MCC results, if only one of the cloud models were utilized

As seen from these results, our evaluations show taking advantage of our prior knowledge of the label of the data helped our proposed model achieve better performance.

5.6 Conclusion and Future Direction

DAI has proven beneficial in applications such as IIoT. In this chapter, we have proposed a low overhead DAI called ADDAI. In our proposed framework, anomaly detection is done close to the sensor level, while more investigation on the data can be done in the cloud.

The produced ADDAI model is able to process most of the data locally with good performance while reducing communication overload. We have achieved this by utilizing a shallow or simple model when there are capability restrictions on the local processing unit. We use more complex deep models in higher level of hierarchy.

To conserve the communication resources, we send a compressed version of the data to the cloud. Through our proposed models, we also prevented the cloud from seeing the actual raw data by sending a latent variant of the data to the cloud. We show empirical proof of performance improvement and decreased communication cost of our proposed technique.

As a future direction, other tasks can be distributed as well. Spanning from simple anomaly detection to explaining the reasons behind the AI's actions. In this chapter, we did not consider different computation powers at the local units. We assumed homogeneous processors. Since not all the processing entities might have equal computational power, the distributed learners could be designed to be feasible with available processing resources, such as computation, memory, time, and communication. Depending on the type of computations, the tasks can be divided among the sensor's platforms, the local processors, and the remote processor.

Chapter 6

Conclusion

Leveraging the internet of things (IoT) technology in the industrial control systems (ICSs), known as the industrial internet of things (IIoT), has caused the industrial revolution in the last decade. IIoTs are mostly mission-critical applications with high-availability requirements. Their operations lead to a huge amount of data that can be easily managed through big data analysis methods, and the modern intelligent security solutions can ensure safe and smooth operations.

About a decade ago, the IoT technology was integrated in industrial fields. Before that, they were considered as a part of operational technology (OT) monitoring and regulating devices. They were isolated from the outside world, hence no outsider threat. They were pre-defined and any out-of-ordinary behavior would be pretty easily spotted, hence no insider threat. The life time of these industrial platforms is several decades, the hardware and the compatible software cannot change very frequently. Therefore, despite the technological advances, we still need to deal with old infrastructure.

Integration of the Internet was inevitable. This caused frequent changes, more sophisticated operations, being open to the outside world, not being able to use pre-defined platforms. Hence, new vulnerabilities from the inside and outside emerged. Utilizing artificial intelligent (AI) and machine learning (ML) to secure the IIoT networks has been proven efficient and effective in the research community. However, they have been hardly, if ever, utilized in real-world industry settings. In this dissertation, we have studied some of the challenges and discussed our endeavors to address them.

The first challenge that we have tackled is data scarcity. Due to privacy concerns, the industries never release their network data to the public. Therefore, we face the lack of specific datasets to train our learning models. Hence, we developed our lab-scale IIoT testbed to conduct cyber-attacks and to design a AI/ML-based intrusion detection systems (IDSs). A popular industrial processor that is widely used in water treatment is replicated. We have included both analog and digital sensors in the testbed to be inclusive of both possible types of data.

In short, the task of this testbed is to keep the water level between two pre-defined levels. At the same time, it measures the turbidity level of the water and illuminates one of the red, yellow or green lights of the turbidity alarm, based on the cloudiness level of the water. Modbus was utilized as the communication protocol in our testbed since it is one of the most popular IIoT protocols commonly used in the industrial control systems. The logic of the PLC is programmed using the Ladder language.

Using our extensive studies, we have built our dataset completely comprehensive such that it would include all the security controls that AI/ML can help detecting. Since attacks that target the "authorization" aspect of security controls have not been studied previously by other research works, we made sure we include this kind of threats (by running backdoor attacks) in our attack scenarios, so that the developed dataset would be comprehensive.

We have deployed backdoor, command injection, DoS, and reconnaissance attacks against the system and demonstrate how an anomaly detection system using learning models can perform in detecting these attacks. We have evaluated the performance through representative metrics to have a fair point of view on the effectiveness of the methods. We also tested out different imbalanced ratios. Our evaluations show that despite in sever cases (i.e., when only 0.1% of the dataset is attack), learning models would fail to learn or detect the attacks at all. It is ironic enough that sever imbalanced cases are actually the cases that happen in real-world settings. We even seek out help from a popular over-sampling technique (SMOTE) but still a little less than half of the attacks stayed undetected.

The second challenged that we address is the black-box nature of the learning models. Not being able to interpret the behavior of the AI, or how it ended up making a decision, would inevitably leave a burden on its utilization in practice especially in sensitive applications. Arguably, the success of learning models come with the cost of high complexity that the human users would not be able to comprehend the logic behind its internal workings.

Explainable AI (XAI) has merged as a popular area in the recent years to open up the black box and reason behind its behaviors. Nowadays, XAI is considered a necessary tool for the next generation of AI. This is not only to explain AI's behavior, but also to be able find out if the model makes mistakes or has been maliciously compromised. Therefore, we have taken advantages of principles of XAI in this dissertation to shed lights on AI's decisions.

However, when it comes to applications that deal with numerical data, there is a significant lack of research. Most of the works done in the XAI area have focused on image applications. At the same time, the ones that are applicable to numerical data are very time consuming or do not provide good performance. Meanwhile, several critical applications such as cybersecurity and IIoT mostly deal with numerical data. Therefore, if we want to seek to the available methods, we would need to sacrifice either the performance, the speed, and/or reusability.

To address this challenged we have proposed a novel XAI called Transparency Relying Upon Statistical Theory (TRUST). By utilizing TRUST on our AI-based security solution, we are able to add transparency, independence, and more importantly gain trust in learning models. Simply put, TRUST models the statistical behavior of the AI's outputs in an AI-based system. We have employed statistical principles to reason different reactions of the AI towards different classes of the data. TRUST accurately estimates the distributions of the AI's outputs and calculates the likelihood of new samples belonging to each class.

One important benefit of TRUST is that it is model agnostic, meaning it can work with any AI models. Most of the developed XAI in the literature are specific to a unique AI/ML model or a group of learners or put restrictions on it (e.g., being differentiable for all the values). Another benefit is that it works as a surrogate model, meaning it works in parallel with the AI model and does not interfere with it. One might argue, why not using an interpretable AI/ML model, so we would not need to use an XAI model? The answer to this question is simple. Performance and complexity (hence no transparency in the model) have an inverse relationship. In sensitive applications, where it is critical to achieve the highest possible performance, we cannot sacrifice how accurate the model performs for interpretability.

We have also demonstrated the superiority of our developed model to a currently popular XAI model, LIME, that can be used for numerical application. By being 25 times faster and more accurate than LIME, TRUST surpassed LIME to be utilized in real-time and critical applications. The success of TRUST in speed comes from the fact that it includes a core model that needs to be developed once and simply applied thereafter for explaining new samples.

Another challenge that we have addressed in this dissertation is the problem of high computational load of AI. In systems like IIoT that low latency is critical, operations need to run smoothly. If an attack happens and it is discovered later, it might affect a large portion of the infrastructure and lead to disastrous outcomes. Hence, the security solutions need to be as fast as the operations themselves. Anomalies should be detected near the source of the data generations, which are the sensors and actuators. Requiring all the data to be offloaded to a central processing would add too much delay. In addition, due to the huge volume of the data, the central unit might become a bottleneck in the system.

To solve this problem, we have utilized distributed AI (DAI). Distributing the AI's computation can be beneficial in applications such as IIoT to speed up the anomaly detection. In this dissertation, we have proposed a low overhead anomaly detection called Anomaly Detection using Distributed AI (ADDAI). In our proposed framework, anomaly detection is done close to the sensor level, while more investigation on the data can be done in the cloud.

The decisions that are made at the local units are called local decisions. Since the local decisions are made using shallow autoencoder models, sometimes we may not trust them and actually want to do more computations in the cloud. We utilize the local decisions in the cloud processing. In the cloud, we have three specific learning models that one is more sensitive on false negatives, another on false positives and a third one that is a neutral model. By defining the thresholds on the reconstructions errors of the samples, we used the local decisions to feed the sample to one of the three cloud models for further investigation.

It is important to mention that, ADDAI is also successful in conserving the communication resources. We send a compressed version of the data to the cloud. We have demonstrated the saving in the communication, which is almost by a half. Another added benefit of our proposed models, is that it preserves the privacy requirements. Since there is no need to send the raw data to the cloud, the privacy of the information are preserved by sending a latent variant of the data to the cloud.

All in all, through this dissertation, we envision a thrive in the research area of network security of IIoT using AI. We hope releasing a tailored dataset to the public, providing a way to reason behind AI's decision, distributing the computations for faster predictions would sparkle more interest in future research works in this domain. And ultimately helping page the way of AI-based security solutions as standalone units in IIoT systems.

References

[1] M. Abadi, A. Chu, I. Goodfellow, B. McMahan, I. Mironov, K. Talwar, and L. Zhang. “Deep learning with differential privacy.” In: *23rd ACM Conference on Computer and Communications Security (ACM CCS)* (2016), pp. 308–318.

[2] A. Adadi and M. Berrada. “Peeking Inside the Black-Box: A Survey on Explainable Artificial Intelligence (XAI).” In: *IEEE Access* 6 (2018), pp. 52138–52160.

[3] Mahbuba Afrin, Jiong Jin, Akhlaqur Rahman, Ashfaqur Rahman, Jiafu Wan, and Ekram Hossain. “Resource Allocation and Service Provisioning in Multi-Agent Cloud Robotics: A Comprehensive Survey.” In: *IEEE Communications Surveys and Tutorials* 23.2 (2021), pp. 842–870.

[4] N. AfzaliSeresht, Q. Liu, and Y. Miao. “An explainable intelligence model for security event analysis.” In: *Australasian Joint Conference on Artificial Intelligence* (2019), pp. 315–327.

[5] M. A. Ahmad, C. Eckert, and A. Teredesai. “Interpretable Machine Learning in Healthcare.” In: *2018 ACM International Conference on Bioinformatics, Computational Biology, and Health Informatics (BCB Ã¢â, ¬â„¢18), Association for Computing Machinery*. New York, NY, 2018, pp. 559–560.

[6] C. Alcaraz. *Security and Privacy Trends in The Industrial Internet of Things*. first. Springer International Publishing, 2019.

[7] B. A. Alejandro et al. “Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI.” In: *Information Fusion* 58 (2020), pp. 82–115.

[8] T. Alves, R. Das, and T. Morris. “Embedding Encryption and Machine Learning Intrusion Prevention Systems on Programmable Logic Controllers.” In: *IEEE Embedded Systems Letters* 10.3 (2018), pp. 99–102.

[9] K. Amarasinghe, K. Kenney, and M. Manic. “Toward Explainable Deep Neural Network Based Anomaly Detection.” In: *11th IEEE International Conference Human System Interaction (HSI)*. Gdansk, 2018, pp. 311–317.

[10] A. Antonini, A. Barenghi, G. Pelosi, and S. Zonouz. "Security challenges in building automation and SCADA." In: *2014 International Carnahan Conference on Security Technology (ICCST)*. Rome, 2014, pp. 1–6.

[11] *Argus*. Available at: `https://qosient.com/argus/`. Date accessed: October 22, 2021.

[12] T. Bartman and K. Carson. "Securing Communications for SCADA and Critical Industrial Systems." In: *69th Annual Conference for Protective Relay Engineers (CPRE)*. College Station, TX, 2016, pp. 1–10.

[13] I. Bartoletti. "AI in Healthcare: Ethical and Privacy Challenges." In: *Artificial Intelligence in Medicine*. Cham: Springer International Publishing, 2019, pp. 7–10.

[14] J. M. Beaver, R. C. Borges-Hink, and M. A. Buckner. "An Evaluation of Machine Learning Methods to Detect Malicious SCADA Communications." In: *12th International Conference on Machine Learning and Applications*. Miami, FL, 2013, pp. 54–59.

[15] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. "SMOTE: Synthetic Minority Over-Sampling Technique." In: *Journal of Artificial Intelligence Research* 16 (2002), pp. 321–357.

[16] J. Chen, K. Li, Q. Deng, K. Li, and P. S. Yu. "Distributed Deep Learning Model for Intelligent Video Surveillance Systems with Edge Computing." In: *IEEE Transactions on Industrial Informatics* (2019).

[17] G. R. Clarke, D. Reynders, and E. Wright. *Practical modern SCADA protocols: DNP3, 60870.5 and related systems*. Newnes, Elsevier, 2004.

[18] K. A.P. da Costa, J. P. Papa, C. O. Lisboa, R. Munoz, and V. H. C. de Albuquerque. "Internet of Things: A Survey on Machine Learning-Based Intrusion Detection Approaches." In: *Computer Networks* (2019), pp. 147–157.

[19] T.M. Cover and J.A. Thomas. *Elements of Information Theory*. second. John Wiley & Sons, Chichester, 2006.

[20] Evren Daglarli. "Explainable Artificial Intelligence (XAI) Approaches and Deep Meta-Learning Models for Cyber-Physical Systems." In: *IGI Global* (2021), pp. 42–67.

[21] S. Dunhaupt. *Vulnerabilities of Industrial Automation Systems*. RUHR Universitat, 2012.

[22] O. Eigner, P. Kreimel, and P. Tavolato. "Detection of Man-in-the-Middle Attacks on Industrial Control Networks." In: *International Conference on Software Security and Assurance (ICSSA)*. St. Polten, Austria, 2016, pp. 64–69.

[23] K. T. Erickson. *Programmable Logic Controllers: An Emphasis on Design and Application*. Dogwood Valley Press, LLC, 2011.

[24] G. Falco, C. Caldera, and H. Shrobe. "IIoT Cybersecurity Risk Modeling for SCADA Systems." In: *IEEE Internet of Things Journal* (2018), pp. 4486–4495.

[25] A. Fisher, C. Rudin, and F. Dominici. "Model Class Reliance: Variable importance measures for any machine learning model class, from the 'Rashomon' perspective." 2018. arXiv: `1801.01489`.

[26] K. Gade, S. C. Geyik, K. Kenthapadi, V. Mithal, and A. Taly. "Explainable AI in Industry." In: *25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD 2019), Association for Computing Machinery*. New York, NY, 2019, pp. 3203–3204.

[27] I. Garcia-Magarino, R. Muttukrishnan, and J. Lloret. "Human-Centric AI for Trustworthy IoT Systems With Explainable Multilayer Perceptrons." In: *IEEE Access* 7 (2019), pp. 125562–125574.

[28] Gartner. *Gartner Says Nearly Half of CIOs Are Planning to Deploy Artificial Intelligence*. Available at: `https://tinyurl.com/5xupkeur`. Date accessed: October 22, 2021. 2018.

[29] S. Gibbs. "Triton: hackers take out safety systems in 'watershed' attack on energy plant." [Online]. Available: [Accessed August 2018].

[30] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. `http://www.deeplearningbook.org`. MIT Press, 2016.

[31] Grand View Research. "Data Annotation Tools Market Size, Share & Trends Analysis Report." In: *Market Analysis Report, Report ID: GVR-2-68038-938-8* (2020).

[32] R. Guidotti, A. Monreale, F. Turini, D. Pedreschi, and F. Giannotti. "A survey of methods for explaining black box models." In: *ACM Computing Surveys* 51.5 (2018), 93:1–93:42.

[33] W. Guo. "Explainable Artificial Intelligence for 6G: Improving Trust between Human and Machine." In: *IEEE Communications Magazine* 58.6 (2020), pp. 39–45.

[34] Y. He, G. J. Mendis, and J. Wei. "Real-Time Detection of False Data Injection Attacks in Smart Grid: A Deep Learning-Based Intelligent Mechanism." In: *IEEE Transactions on Smart Grid* 8.5 (2017), pp. 2505–2516.

[35] *HiveMQ*. Available at: `https://www.hivemq.com/tags/mqtt-security-fundamentals/`. Date accessed: October 22, 2021.

[36] D. G. Holmberg. *BACnet Wide Area Network Security Threat Assessment*. National Institute of Standard and Technology, 2003.

[37] A. Holzinger, C. Biemann, C. S. Pattichis, and D. B. Kell. "What Do We Need to Build Explainable AI Systems for the Medical Domain?" In: *arXiv preprint arXiv: 1712.09923* (2017).

[38] *IEEE Standards Association*. Available at: `https://standards.ieee.org/standard/1815-2012.html`. Date accessed: October 22, 2021.

[39] R. Jain. *The Art of Computer Systems Performance Analysis*. Wiley Interscience, 1991.

[40] J. Jayasamraj. *SCADA Communication & Protocols*. Power System Operation Corporation Ltd, 2013.

[41] *Kali Linux*. Available at: `https://www.kali.org`. Date accessed: October 22, 2021.

[42] A. Kassambara. *Practical Guide To Principal Component Methods in R*. STHDA, 2017.

[43] A. Keliris, H. Salehghaffari, B. Cairl, P. Krishnamurthy, M. Maniatakos, and F. Khorrami. "Machine learning-based defense against process-aware attacks on Industrial Control Systems." In: *IEEE International Test Conference (ITC)*. Fort Worth, TX, 2016, pp. 2505–2516.

[44] *Keras*. Available at: `https://keras.io`. Date accessed: October 22, 2021.

[45] K. Kim, J. Lynskey, S. Kang, and C.S. Hong. "Prediction Based Sub-Task Offloading in Mobile Edge Computing." In: *2019 International Conference on Information Networking (ICOIN)*. 2019, pp. 448–452.

[46] C. Li, C. Wang, and Y. Luo. "An efficient scheduling optimization strategy for improving consistency maintenance in edge cloud environment." In: *The Journal of Supercomputing* 76 (2020), pp. 6941–6968.

[47] J. Li, K. Cheng, S. Wang, F. Morstatter, R. P. Trevino, J. Tang, and H. Liu. "Feature selection: A data perspective." In: *ACM Computing Surveys* 15 (2017), 94:1–94:45.

[48] Z. Liu, B. Xiao, M. Alrabeiah, K. Wang, and J. Chen. "Single image dehazing with a generic model-agnostic convolutional neural network." In: *IEEE Signal Processing Letters* 26.6 (2019), pp. 833–837.

[49] S. M. Lundberg and S. I. Lee. "A unified approach to interpreting model predictions." In: *Advances in Neural Information Processing Systems 30*. Ed. by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Curran Associates, Inc., 2017, pp. 4765–4774.

[50] L. A. Maglaras and J. Jiang. "Intrusion detection in SCADA systems using machine learning techniques." In: *Science and Information Conference*. London, U.K., 2014, pp. 626–631.

[51] S. Mahamood. "Explainable Artificial Intelligence and its potential within Industry." In: *Proceedings of the 1st Workshop on Interactive Natural Language Technology for Explainable Artificial Intelligence (NL4XAI 2019)*. Vol. 58. 2019, pp. 1–2.

[52] M. Mantere, M. Sailio, and S. Noponen. "Feature Selection for Machine Learning Based Anomaly Detection in Industrial Control System Networks." In: *IEEE International Conference on Green Computing and Communications*. Besancon, France, 2012, pp. 771–774.

[53] D. Marino, C. Wikramasinghe, and M. Manic. "An Adversarial Approach for Explainable AI in Intrusion Detection Systems." In: *44th Annual Conference of the IEEE Industrial Electronics Society, IECON 2018*. Washington DC, 2018.

[54] S. M. Mathews. "Explainable artificial intelligence applications in NLP, biomedical, and malware classification: A literature review." In: *Intelligent Computing-Proceedings of the Computing Conference* (2019), pp. 1269–1292.

[55] S. Maxwell, H. Delaney, and K. Kelley. *Designing Experiments and Analyzing Data: A Model Comparison Perspective*. third. Routledge, 2017.

[56] A. Mijuskovic, A. Chiumento, R. Bemthuis, A. Aldea, and P. Havinga. "Resource Management Techniques for Cloud/Fog and Edge Computing: An Evaluation Framework and Classification." In: *Sensors* (2021).

[57] T. Miller. "Explanation in Artificial Intelligence: Insights From the Social Sciences." In: *Artificial Intelligence* 267 (2019), pp. 1–38.

[58] C. Molnar. *Interpretable machine learning. A Guide for Making Black Box Models Explainable.* Available at: `https://christophm.github.io/interpretable-ml-book/`. Date accessed: October 22, 2021. 2018.

[59] C. Molnar. *Interpretable machine learning. A guide for making black box models explainable.* Available at: `https://christophm.github.io/interpretable-ml-book/`. Date accessed: October 22, 2021. 2018.

[60] T. Morris and W. Gao. "Industrial Control System Traffic Data Sets for Intrusion Detection Research." In: *International Conference on Critical Infrastructure Protection*. Berlin, Heidelberg, 2014.

[61] N. Moustafa and J. Slay. "UNSW-NB15: a Comprehensive Data Set for Network Intrusion Detection Systems (UNSW-NB15 Network Data Set)." In: *2015 Military Communications and Information Systems Conference (MilCIS)*. Canberra, ACT, 2015, pp. 1–6.

[62] *MQTT*. Available at: `http://mqtt.org`. Date accessed: October 22, 2021.

[63] S. K. Ng, T. Krishnan, and G. J. McLachlan. "The EM algorithm." In: *Handbook of computational statistics*. Springer, 2012, pp. 139–172.

[64] C. Oxborough and E. Cameron. *Explainable AI: driving business value through greater understanding*. Available at: `https://www.pwc.co.uk/xai`. Date accessed: October 22, 2021. 2018.

[65] J. Pages. *Multiple Factor Analysis by Example Using R*. New York: Chapman and Hall/CRC, 2015.

[66] A. Paszke et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library." In: *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035.

[67] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python." In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[68] S. Ponomarev and A. E. Voronkov. "Multi-agent systems and decentralized artificial superintelligence." In: *arXiv:1702.08529* (2017).

[69] S. Potluri, N. F. Henry, and C. Diedrich. "Evaluation of Hybrid Deep Learning Techniques for Ensuring Security in Networked Control Systems." In: *22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*. Limassol, Cyprus, 2017, pp. 1–8.

[70] C. Promper, D. Engel, and R. C. Green. "Anomaly detection in smart grids with imbalanced data methods." In: *IEEE Symposium Series on Computational Intelligence (SSCI)*. Honolulu, HI, 2017.

[71] J. Queiroz, P. Leitão, J. Barbosa, and E. Oliveira. "Distributing Intelligence among Cloud, Fog and Edge in Industrial Cyber-physical Systems." In: *ICINCO*. 2019.

[72] J. Rabold, H. Deininger, M. Siebers, and U. Schmid. "Enriching visual with verbal explanations for relational concepts - combining LIME with Aleph." In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer. 2019, pp. 180–192.

[73] M. T. Ribeiro, S. Singh, and C. Guestrin. "Why Should I Trust You?? Explaining the Predictions of Any Classifier." In: *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*. San Francisco, CA, 2016, pp. 1135–1144.

[74] M.T. Ribeiro, S. Singh, and C. Guestrin. "Anchors: high-precision model-agnostic explanations." In: *Proceedings of the AAAI Conference on Artificial Intelligence* 32.1 (2018).

[75] O. Sagi and L. Rokach. "Explainable decision forest: Transforming a decision forest into an interpretable tree." In: *Information Fusion* 61 (2020), pp. 124–138.

[76] S. Samtani, S. Yu, H. Zhu, M. Patton, and H. Chen. "Identifying SCADA Vulnerabilities Using Passive and Active Vulnerability Assessment Techniques." In: *2016 IEEE Conference on Intelligence and Security Informatics (ISI)*. Tucson, AZ, 2016, pp. 25–30.

[77] *Schneider PLC, M241CE40*. Available at: `http://www.filkab.com/files/category_files/file_3073_Bg.pdf`. Date accessed: October 22, 2021.

[78] *scikit-learn*. Available at: `https://scikit-learn.org/stable/`. Date accessed: October 22, 2021.

[79] S. Shekarforoush, R. Green, and R. Dyer. "Classifying commit messages: A case study in resampling techniques." In: *International Joint Conference on Neural Networks (IJCNN)*. Anchorage, AK, 2017, pp. 1273–1280.

[80] I. A. Siddavatam, S. Satish, W. Mahesh, and F. Kazi. "An Ensemble Learning for Anomaly Identification in SCADA System." In: *7th International Conference on Power Systems (ICPS)*. Pune, India, 2017, pp. 457–462.

[81] I. Siniosoglou, P. Radoglou-Grammatikis, G. Efstathopoulos, P. Fouliras, and P. Sarigiannidis. “A Unified Deep Learning Anomaly Detection and Classification Approach for Smart Grid Environments.” In: *IEEE Transactions on Network and Service Management* 18.2 (2021), pp. 1137–1151.

[82] T. Skripcak and P. Tanuska. “Utilisation of on-line machine learning for SCADA system alarms forecasting.” In: *Science and Information Conference*. London, U.K., 2013, pp. 477–484.

[83] Stanford. “Artificial Intelligence Index Annual Report.” In: (2019).

[84] P. Stone and M. Veloso. “Multiagent systems: a survey from a machine learning perspective.” In: *Autonomous Robots* (2000), pp. 345–383.

[85] K. Stouffer, V. Pillitteri, S. Lightman, M. Abrams, and A. Hahn. *Guide to Industrial Control Systems (ICS) Security*. National Institute of Standards and Technology, 2015.

[86] K. Stouffer, V. Pillitteri, S. Lightman, M. Abrams, and A. Hahn. “Guide to Industrial Control Systems (ICS) Security.” National Institute of Standards and Technology, [Online]. Available: [Accessed August 2018].

[87] D. Sui, Y. Chen, J. Zhao, Y. Jia, Y. Xie, and W. Sun. “FedED: Federated Learning via Ensemble Distillation for Medical Relation Extraction.” In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2020, pp. 2118–2128.

[88] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani. “A Detailed Analysis of the KDD CUP 99 Data Set.” In: *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*. Ottawa, ON, 2009, pp. 1–6.

[89] M. A. Teixeira, T. Salman, M. Zolanvari, R. Jain, N. Meskin, and M. Samaka. “SCADA System Testbed for Cybersecurity Research Using Machine Learning Approach.” In: *Future Internet* 10 (2018), pp. 1–15.

[90] E. Tjoa and C. Guan. “A Survey on Explainable Artificial Intelligence (XAI): Toward Medical XAI.” In: *IEEE Transactions on Neural Networks and Learning Systems* (2020), pp. 1–21.

[91] K. Tocze and S. Nadjm-Tehrani. “A taxonomy for management and optimization of multiple resources in edge computing.” In: *Wireless Communications and Mobile Computing* (2018).

[92] I. Ullah and Q. H. Mahmoud. “A Hybrid Model for Anomaly-based Intrusion Detection in SCADA Networks.” In: *IEEE International Conference on Big Data (Big Data)*. Boston, MA, 2017, pp. 2160–2167.

[93] J. Verba and M. Milvich. “Idaho National Laboratory Supervisory Control and Data Acquisition Intrusion Detection System (SCADA IDS).” In: *IEEE Conference on Technologies for Homeland Security*. Waltham, MA, 2008, pp. 469–473.

[94] M. Wang, K. Zheng, Y. Yang, and X. Wang. "An Explainable Machine Learning Framework for Intrusion Detection Systems." In: *IEEE Access* 8 (2020), pp. 73127–73141.

[95] Webroot. *Game Changers: AI and Machine Learning in Cybersecurity*. Available at: `https://tinyurl.com/49sf724t`. Date accessed: October 22, 2021. 2017.

[96] "Wikipedia."

[97] *Wireshark*. Available at: `https://www.wireshark.org/`. Date accessed: October 22, 2021.

[98] H. Wu, W. Knottenbelt, K. Wolter, and Y. Sun. "An Optimal Offloading Partitioning Algorithm in Mobile Computing, Quantitative Evaluation of Systems." In: *13th International Conference, QEST 2016*. Quebec City, Canada, 2016, pp. 311–328.

[99] E. P. Xing, Q. Ho, P. Xie, and D. Wei. "Strategies and principles of distributed machine learning on big data." In: *Engineering 2* (2016), pp. 179–195.

[100] Y. Xu, Y. Yang, T. Li, J. Ju, and Q. Wang. "Review on cyber vulnerabilities of communication protocols in industrial control systems." In: *2017 IEEE Conference on Energy Internet and Energy System Integration (EI2)*. Beijing, 2017, pp. 1–6.

[101] S. Yasakethu and J. Jiang. "Intrusion Detection via Machine Learning for SCADA System Protection." In: *1st International Symposium on ICS & SCADA Cyber Security Research (ICS-CSR)*. 2013, pp. 101–105.

[102] C. Yu, H. Tang, C. Renggli, S. Kassing, A. Singla, D. Alistarh, C. Zhang, and J. Liu. "Distributed learning over unreliable networks." In: *Proceedings of the 36th International Conference on Machine Learning, Long Beach, California, PMLR 97* (2019).

[103] M.R. Zafar and N.M. Khan. "DLIME: a deterministic local interpretable model-agnostic explanations approach for computer-aided diagnosis systems." In: *arXiv preprint arXiv:1906.10263* (2019).

[104] Y. Zhang, M. D. Ilic, and O. K. Tonguz. "Mitigating Blackouts via Smart Relays: A Machine Learning Approach." In: *The IEEE* 99.1 (Jan. 2011), pp. 94–118.

[105] B. Zhu, A. Joseph, and S. Sastry. "A Taxonomy of Cyber Attacks on SCADA Systems." In: *2011 International Conference on Internet of Things and 4th International Conference on Cyber*. Dalian: Physical and Social Computing, 2011, pp. 380–388.

[106] M. Zolanvari, A. Ghubaish, and R. Jain. "ADDAI: Anomaly Detection using Distributed AI." In: *IEEE ICNSC (International Conference on Networking, Sensing and Control)*. China, 2021.

[107] M. Zolanvari, M. A. Teixeira, L. Gupta, K. M. Khan, and R. Jain. "Machine learning-based network vulnerability analysis of Industrial Internet of Things." In: *IEEE Internet of Things Journal* 6 (2019), pp. 6822–6834.

[108] M. Zolanvari, M. A. Teixeira, and R. Jain. “Effect of Imbalanced Datasets on Security of Industrial IoT Using Machine Learning.” In: *IEEE Intelligence and Security Informatics (ISI)*. Miami, FL, 2018, pp. 112–117.

[109] M. Zolanvari, Z. Yang, K. Khan, R. Jain, and N. Meskin. “TRUST XAI: A Novel Model for Explainable AI with An Example Using IIoT Security.” In: *IEEE Internet of Things Journal* (2021).