Using machine learning to detect intrusions in industrial control systems

Jacob Nicolai Larsen

Thesis submitted for the degree of master's in information security

60 credits

Faculty of informatics

Faculty of mathematics and natural sciences

University of Oslo

Spring 2022

Investigating network anomaly detection methods in industrial control systems

Comparing machine learning to detect intrusions in control- and traditional networks

Jacob Nicolai Larsen

Abstract

In recent decades ICS systems have been a vulnerable target for cybercriminals. Cyber attacks such as Stuxnet show the vulnerabilities in communication between Supervisory data and Aquiasions (SCADA) systems and Programmable Logic Controllers (PLC) through unknown attack vectors or mutations of old ones. To protect against these unknown attack vectors are Intrusion Detection Systems (IDS). A Networks-based IDS has the potential to safeguard against these unknown attack vectors by detecting anomalies in the network traffic through Artificial Intelligence (AI). Branching out from AI are Machine Learning (ML) and the newer Deep Learning (DL). Researchers have extensively researched anomaly NIDS in traditional networks through ML and DL. Despite the comprehensive analysis, IDS still faces challenges in improving performance for detecting new attack vectors. As SCADA networks are more isolated than conventional networks, we propose that machine learning is better suited to detect anomalies in these environments. Derived from this experiment is the question if the newer DL methods perform better than ML methods.

The thesis explores this by going through the fundamentals of ICS, such as devices communicating and their protocols. IDS are analyzed to understand better where implementation of AI. We discover previous research on AI solutions for traditional and SCADA networks before selecting AI methods to compare both types of network traffic.

The resulting literature study led to us applying two DL methods, long short-term memory (LSTM) and Autoencoder (AE), to NSL-KDD and Electra Modbus. In we used ML methods, Random Forest (RF), Isolation Forest (IS), Support-vector machine (SVM), One-Class SVM (OCSVM), and Artificial Neural Network (ANN) to the traditional network.

The study showed significantly better results on the SCADA network traffic. However, as the research shows, SCADA systems have no benchmark dataset for network traffic. With no benchmark dataset, it is difficult to conclude if the learner performs better. By mainly focusing on the SCADA system, there have been some biases for the high result of the learners. In the future, we require a more structured approach to comparing learners of SCADA and traditional network traffic.

Preface

This project was carried out as the final part of my master’s degree in information security at the University of Oslo.

First and foremost, I thank my supervisor, Janne Merete Hagen, for her guidance, support, encouragement, and pleasant conversations through the writing of this thesis.

I would also like to thank my family, friends, and girlfriend Margrethe for their continued love and support.

Jacob Nicolai Larsen
University of Oslo, June 2022

Content

List of figures:

List of tables:

List of equations:

1 Introduction

In 2010 the Iranian nuclear program experienced huge setbacks as the gas centrifuges separating atomic material were spinning out of control. After a thorough investigation, security analysts uncovered one of the most sophisticated malicious computer worms ever created. Stuxnet was the resulting worm of a 5-year project exploiting unknown vulnerabilities (Baezner & Robin, 2017). Analyzing the worm showed it targeted the supervisory control and acquisition (SCADA) system that communicates with and controls the programmable logic controller (PLC) controlling the centrifuges (Baezner & Robin, 2017). Stuxnet establishes the potential objective of hacking these systems using unknown vulnerabilities. Because of Stuxnet, there has been an increase in attacks on ICS (Knapp & Langill, 2014).

Intrusion Detection Systems (IDS) and, more specifically, Network-based Intrusion Detection Systems (NIDS) provide this desired security by constantly monitoring the network traffic between devices (Zeeshan, Adnan, Cheah, Johari, & Farhan, 2021). By integrating Artificial Intelligence (AI) and its subcategories Machine Learning (ML) and Deep Learning (DL), pattern from the network can be learned and used to detect these unknown attack vectors.

There has been done extensive research on ML and DL as NIDS. Both ML and DL are powerful tools for learning valuable features from the network traffic and predicting normal and abnormal activities based on the learned patterns (Zeeshan, Adnan, Cheah, Johari, & Farhan, 2021). IDS still faces challenges in improving detection accuracy while reducing false alarm rates and detecting novel intrusions (Zeeshan, Adnan, Cheah, Johari, & Farhan, 2021).

There are many research papers on NIDS on traditional networks. The research on NIDS in SCADA networks is not quite as extensive (Alimi, Ouahada, Abu-Mahfouz, Rimer, & Alimi, 2021). The difference in this thesis compared to other studies on the topic is comparing the DL and ML models to discover if AI is more suitable in SCADA networks, as SCADA devices communicate in a more controlled environment than traditional business networks (Knapp & Langill, 2014).

2 Aim of thesis

This thesis aims to apply deep learning methodologies to industrial networks, comparing the same models on traditional networks. To accomplish this, the thesis will 1. Do a literature study on ICS understanding the environment, such as the network structure, devices used, and network topologies (Chapter 3-4). 2. Explore IDS to get a high picture view of where machine learning is applied (Chapter 5). 3. Study previous AI solutions in solutions in the literature to select appropriate models (Chapter 6). Do a simulated study applying deep learning methods to both datasets and comparing the results.

2.1 Problem statement

There are two research questions explored in this thesis:

> *Research question 1(RQ1): Is it more suitable to apply machine learning in industrial networks than in traditional networks?*
>
> *Research question 2(RQ2): Is deep learning more powerful in detection intrusions than traditional machine learning methods?*

2.2 Contribution

This thesis contributes a novel approach to comparing AI NIDS for network traffic. With this contribution, future researchers can decide if a heavier focus should be on more isolated networks.

2.3 Structure

Chapter one contains the introduction of the thesis.

Chapter two presents the thesis's problem aim, problem statement, contribution

Chapter three contains the fundamentals of Industrial Control systems used to understand the current situations of the devices that use control networks. Previous devastating attacks, as well as typical attacks, are presented to thoroughly understand the attacks that are present in control network datasets.

Chapter four contains how Industrial Control Systems use networks to communicate. This chapter presents this, such as different communications protocols and general network topologies. Protocols and network topologies give an overview of the data that can be present in control network datasets.

Chapter five contains the fundamentals of Intrusion Detection Systems. This chapter presents current solutions to network attacks and an overview of where the machine learning algorithms reside.

Chapter six presents machine learning algorithms used by the Intrusion Detection System. This chapter explains different learning methodologies and commonly used machine learning algorithms for Intrusion Detection. This chapter also differentiates between machine learning and deep learning methodologies.

Chapter seven presents the methodology used to apply machine learning algorithms. Datasets are selected. One of the datasets contains data from the Control network traffic and one from the traditional network traffic. The attributes of these datasets are analyzed and preprocessed for the use of the selected machine learning models.

Chapter eight presents the results of the machine learning algorithms. This chapter selects the different machine learning hyper-paramet0065rs for bough datasets. For consistency, hyper-parameters for machine learning algorithms are the same to be comparable between the two chosen datasets. The differences between the datasets are also discussed as this directly impacts the performance of the machine learning algorithms. In addition, there is a realistic evaluation of the Control network dataset as this is not a benchmark dataset.

Chapter nine concludes the thesis.

Chapter ten goes through future work to compare traditional and SCADA network

3 Industrial Control System fundamentals

Before being able to apply deep learning models to defend ICS, Sommer & Paxson underly the importance of understanding the data. Starting with what ICS is. Knapp & Langill describes it as a “broad class of automation systems used to provide control and monitoring functionality in manufacturing and industrial facilities” (Knapp & Langill, 2014). The ICS comprises several devices for automation and supervising these processes. The ICS controls the operations of field components such as sensors, actuators, motor drives, gauges, and indicators. In other words, ICS is in practice for industrial automation. Further, it is essential to comprehend commonly used ICS components and how they communicate within the network to protect the industrial network.

3.1 Supervisory Control and Data Acquisition (SCADA)

Supervisory Control and Data Acquisition (SCADA) system is typically an automation control system used in different industries such as energy, petroleum, water, and power. The system can monitor entire complexes of plants providing the user with remote and centralized control for any given system. A typical SCADA system contains one or more control center(s) along with several distributed field devices such as 1) Programmable Logic controllers (PLCs), 2) Remote terminal units (RTUs), and 3) intelligent electronic devices (IEDs). The 4) human machine interface (HMI) to communicate and control these field devices. To store the data collected from these devices are the 5) Data historian.

3.2 Programmable Logic Controller (PLC)

Programmable Logic Controllers (PLCs) are specialized industrial computers that automate electromechanical processes (Qassim, Jamil, Patel, & Ja'affar, 2019). PLCs do not use an operating system (OS) to perform their tasks compared to desktop computers. Instead, they have specific programs to generate output actions responding to a particular input keeping the data transmitted to a minimum. To function in production environments, the physical design of PLCs is robust. The program logic from the PLCs is kept simple as they often are used for real-time processing. For the simplicity of the program logic, the PLC uses ladder logic or sequential function carts to operate. PLCs essentially perform an output action based on the state of the input. To communicate, PLCs can use various communication protocols but typically uses Modbus, ControlNet, EtherNet/IP, PROFIBUS, or PROFINET. Industrial Protocols are discussed further in the Industrial network chapter. However, Independent of the protocol in use, the end goal of the PLC is to automate processes by checking input, applying logic, and adjusting the output accordingly (Knapp & Langill, 2014).

3.3 Remote Terminal Unit (RTU)

Another device that monitors and controls the field devices is the RTU. The PLC and the RTU functionality overlap in many areas, and the RTU is often indistinguishable from the PLC (Knapp & Langill, 2014). Compared to the PLC, the RTU resides in remote locations such as the outside. The RTU transmits field parameters from the remote location and sends data back to a master terminal unit (MTU) or directly to a Human Machine Interface (HMI). Because of the remote location, RTUs must be even more robust than PLCs as they must withstand environmental factors such as humidity and temperature. Another problem with remote installation is the requirement for electricity. Local solar power can power the RTUs to accommodate this problem.

3.4 Intelligent Electronic Devices (IED)

The IED is very similar to the PLC and RTU. The difference between the IED, PLC and RTU often overlap, but the IDE usually supports more specific functions, whereas PLC and RTUs have more general use (Knapp & Langill, 2014). Specifically, in many control systems, power grids may disrupt the RTU and PLC because of the high voltage present. IEDs reside in these environments.

3.5 Human machine interface (HMI)

The Human machine interface is the device used to control the PLC, RTU, and IED. The HMI allows operators to interact with the control processes using modern software. In contrast to the above devices, the HMI uses operating systems such as Windows 7 to perform its tasks. The user interface of the HMI display information about how the processes are performing by displaying sensor reading, output from PLCs, and other measurements. Communication to the ICS server happens through protocols such as EtherNet/IP or Modbus (Knapp & Langill, 2014). We discuss these in the industrial protocols chapter.

3.6 Data Historian

For data collection, the SCADA system utilizes a data historian. The historian collects the data through various protocols such as Modbus, PROFIBUS, DNP3, and OPC (Knapp & Langill, 2014). The data the historian logs are multiple types from the control system, such as alarms and events, and stored in a purpose-built database as “tags.”

3.7 SCADA Operations

PLCs and other field devices have a preprogrammed logic where the input of the device affects the output. For the field components to work together autonomously, the SCADA system is on a low level comprised of multiple control cycles or loops. The control loop can, for example, be if the temperature is 90 degrees do action. More complex actions require numerous control loops. On a higher level comes control processes. The control process uses multiple control loops to act and achieve an objective, such as producing an item. The HMI controls each of these processes. Typically, the data from the field components, such as readings, are recorded in an analog format which is translated to digital data and sent to the field device. The field device further transmits the data to an MTU that keeps track of the state of the data with the associated device (Yadav & Paul, 2020). Data is passed along from the MTU to an HMI.

3.8 SCADA communication evolution

The communication of SCADA devices has evolved through many generations. Yadav & Paul, 2020 categorizes these into four eras.

The first generation of SCADA was a Monolithic SCADA system. The monolithic SCADA system was an isolated environment where it could not communicate with other systems as the communication could only happen with vendor-specific protocols. The system RTUs and MTUs would communicate with each other using Wide Area Networks (WAN), which were in an early stage.

Second-generation SCADA was the transition from a monolithic system to a distributed system. Here the use of local area networks (LAN) was used for communication replacing the WAN communication between RTUs and MTUs. As there was no outside communication, security was not a concern at the time (Yadav & Paul, 2020).

Third-generation SCADA, also known as the modern SCADA system, utilizes networks and the web for more cost-efficient solutions and the potential for distributed solutions. Modern SCADA integrates the Internet protocol (IP) based on WAN to communicate, giving the ability to connect to the distributed system. Giving the advantage of accessing the business network from a remote location has enabled products of SCADA system to rise.

Lastly, the fourth generation of SCADA, which is now on the rise, has been integrating the internet of things (IoT) innovations such as Cloud computing reducing integration and deployment costs (Yadav & Paul, 2020).

As seen from the above evolution of SCADA, there have been many changes in how to communicate. There have also been changes in many protocols used. Many utilizing the SCADA system have not changed devices with the rapid changes. Because of this and the long lifespans of SCADA components, a wide variety of communications protocols and legacy systems are in place.

3.9 Vulnerabilitets in SCADA

The numerous legacy systems and modern SCADA using the web and IP-based communication have opened an attack surface that an adversary can exploit, leading to high-level threats. The impact of these cyberattacks can range from disrupting or damaging critical infrastructural operations to causing significant economic losses or, even more dangerously, claiming human lives (Qassim, Jamil, Patel, & Ja'affar, 2019). Recent studies have shown over a million ICS/SCADA systems are connected to the internet with unique IP addresses. Adversaries can effortlessly detect these systems using search engines like Shodan (Babu, Ijyas, P, & Varghese, 2017). In combination, many SCADA protocols are legacy and have not adopted modern encryption in an adequate matter (Babu, Ijyas, P, & Varghese, 2017) should indicate that the need for security is crucial. Gonzalez, Alhenaki, & Mirakhorli have also documented attacks have exponentially increased from a few incidents to hundreds per year. Most adversaries targeted the human-machine interfaces, configurations, and PCSs (Gonzalez, Alhenaki, & Mirakhorli, 2019). There are many vulnerabilities in SCADA for the different assets. This thesis will mainly focus on the vulnerabilities of the control network and communication between field devices, whereby machine learning might detect these attacks early.

3.9.1 Attack categories

Wang & Foo has divided the different types of attacks on the control network into eleven categories. Attacks on control networks can be classified even further, but this thesis will use the attacks described her base for the available attacks on ICS networks. Table 1 describes these attacks.

Table 1: Classification of attack types on control networks (Wang & Foo, 2018).

Attack Type	Description
Analysis	Often occurs at the beginning of a multi-stage attack. The adversary tries to get information about the network, such as IP- and Port-scanning
Reconnaissance	The adversary's main objective is to gather network information for further attacks. Hard to detect
Masquerading	Adversary pretends to be an existing service in the network. Attacks include ARP poisoning and Man in the middle (MITM) attacks. Hard to detect
Replay	Replay attacks disrupt the system flow by sending previously valid packets.
Real-time message modification	Real-time message modifications are almost the same as replay attacks. The difference is the adversary edits real-time packets to achieve a specific goal. Often done when the adversary is MITM
Deneil of service (DoS)	Adversaries perform DoS attacks to disrupt the availability of a service. DoS attacks are accomplished by flooding the service with enough messages so the service can’t handle everything.
Anomalous message injection	In Anomalous message injection, adversaries place malicious content into the legitimate message. The target is infected when accepting or running the malicious contents.
Time delay Attack	Time delay attacks happen after masquerading. The adversary purposely delays legit messages.
Authentication bypass	With authentication bypass, the adversary stealthily gains unauthorized access to the system with high-level access.
Firmware modification	Adversaries launch a counterfeit update to modify the firmware
Worms	After infecting a system, worms replicate and spread to other hosts on the network

3.10 Cyber kill chain

The attack described in table 1 does not often happen in a random order, and for a NIDS to detect these attacks, data used for the machine learning model must contain attacks from different stages. The cyber kill chain (Huntchins, Cloppert, & Amin, 2011) is a model describing an adversary's steps to achieve its objective. The adversary must go through seven steps in the kill chain to complete its purpose. The kill chain is essential when selecting a dataset with appropriate attack steps. 1. Reconnaissance 2. Weaponize 3. Delivery 4. Exploitation 5. Installation 6. Command and control (C2), and 6. Actions on objectives (Hutchins, Cloppert, & Amin, 2011). Stopping the adversary at any of these steps will mitigate the attack.

1. The Reconnaissance step is where the adversary finds information about the target. These attacks can be low frequency and hard to notice.
2. After collecting information about the target, the adversary creates a means of how to deliver the attack.
3. Delivery adversary delivers the malicious code to the vulnerable system.
4. After the malicious code is delivered, the adversary starts exploiting the system.
5. Installation. The installation step creates a persistent threat to the vulnerable system, such as a backdoor. The adversary ensures access to the system for later attacks and persistence.
6. With the command-and-control step, the adversary typically has "hands on the keyboard" access inside the target environment
7. Only after the six previous steps does the adversary achieve its original goal. When defending an ICS network, the main objective is the integrity of network packets. In the Stuxnet case, adversaries compromised the integrity of the traffic packets on the control network. However, adversaries can also ruin the availability of the provided service or expose the confidentiality of Historian data.

4 Industrial networks

We divide Industrial networks into business-, supervisory (production)- and control networks. However, most network design is not this simple, and there is often overlapping in terms of protocols as production supervisory network architecture has integrated Ethernet/IP. This thesis will not focus on the business network part of the industrial network architecture as the SCADA system operates on the supervisory- and control-network segments.

4.1 Industrial network topology

The industrial network (supervisory- and control- network) topology must support many communication protocols for the various connected devices. The supervisory network segment handles communication, such as connecting historians to the ICS server and connecting the ICS to other suppliers. The control network, which is the essential part of the network for this thesis, handles communication, such as sending data from PLC to the historian or HMI.

The most common for an industrial network is bus and ring topologies; to support these, star, tree, and mesh are used (Knapp & Langill, 2014). Figure 1 shows examples of the various topologies.

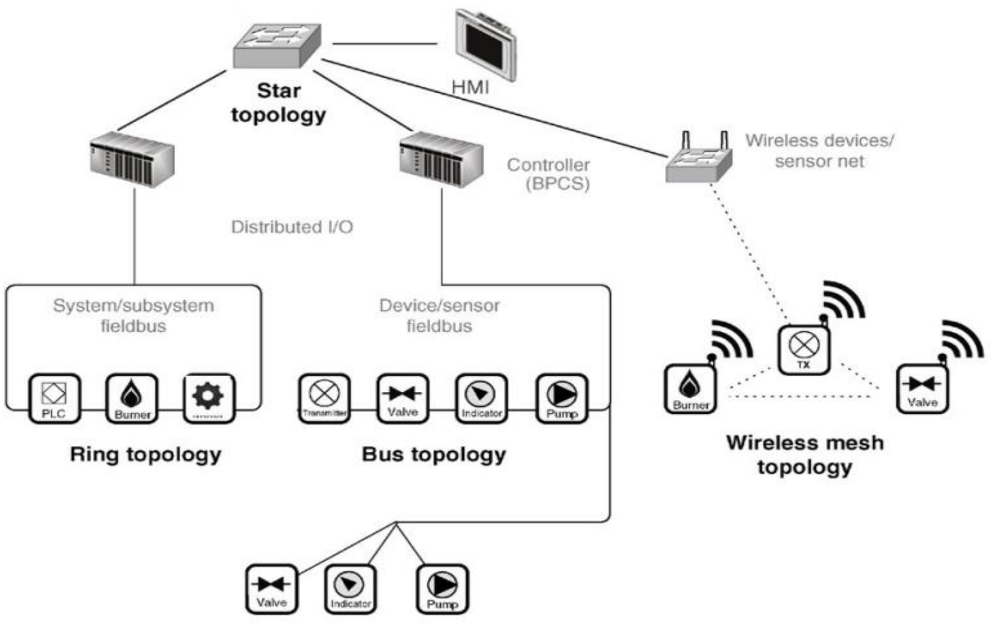


Figure 1: Common network topologies in an industrial network (Knapp & Langill, 2014)

4.2 OSI Model

We first need to look at the OSI reference model to understand the underlying communication protocols used in industrial networks. Networks are complex with many pieces. Network engineers developed the layered protocol stack to acknowledge the complexity. The layered protocol stack, also called Open Systems Interconnection model (OSI model), aggregates functionality/services and protocols into a layered model. Each layer contains the necessary protocols to apply more complex services the further up the model traversed. Industrial network protocols use the OSI model as well as traditional networks. The OSI models break down the functionality needed when sending data between devices. The layers are physical-, data link-, network-, transport-, session-, presentation- and application-layer. Table 2 displays the OSI reference model.

Table 2: OSI reference model

OSI 7 Layer model	Data name
7 Application layer	Message
6 Presentation layer	
5 Session layer	
4 Transport layer	Segment
3 Networking layer	Packet
2 Data link layer	Frame
1 Physical layer	Bits

4.3 Industrial network protocols

SCADA has evolved over many generations, causing various communication protocols to be developed and used. Many of these protocols are old but still in use as the devices have a long life expectancy. As mentioned above, we segmented the industrial network in to supervisory and control networks. Both of these segments use different protocols. Communication protocols used by the supervisory network include protocols such as

Open Process Communications (OPC) and the Inter-Control Center Protocol (ICCP) (Knapp & Langill, 2014). In the control network, protocols used for communication are generally called Fieldbus protocols. Modbus is the oldest and most widespread Fieldbus communication protocol (Collantes & Padilla, 2015), and up to 90% of field devices can use this for communication (ZAREI, 2020). Other protocols found in the field devices are CIP, S7comm, RS-232 and RS-485, DNP3, HART, PROFIBUS and PROFINET, FOUNDATION Fieldbus, BACnet, and more. All the protocols mentioned have been designed to work in the industrial environment but have evolved support to the ever-increasing popularity of IP (Malviya, 2019). Protocols found in ICS datasets relevant to this thesis are Modbus, S7comm, CIP, and HSPA.

4.4 Modbus protocol

The Modbus protocol is a communication protocol published in 1979 and became the unofficial industry standard for communication between PCLs and other devices. The Modbus protocol is easily deployed and only requires a few data transmitting requirements. The only requirements are the packet size of the transmitted data. The Modbus protocol has different options in the transport layer, including using a serial port, ethernet, or the internet protocol suite, commonly known as TCP/IP. Modbus support multiple communications in devices connected to the same cable or on the same ethernet connection. It is typical for the Modbus protocol to be implemented between RTUs and a plant/system supervisory computer in the SCADA system in the power industry. Since 1979, multiple versions of Modbus have been developed, such as RTU, serial, and TCP/IP.

The Modbus protocol uses a client/server architecture. Because of this, it is always the client that initiates the communication. The client initiates the communication by sending a query message while the server responds with the requested data. Figure 2 show this process. The Modbus client query includes the device address where to send the request, a function code defining the request action, any data transmitted, and an error checking field. The server responds with the same function code as the query, the response data, and an error code if the device could not understand the request. (MODICON, Inc., Industrial Automation Systems, 1996).

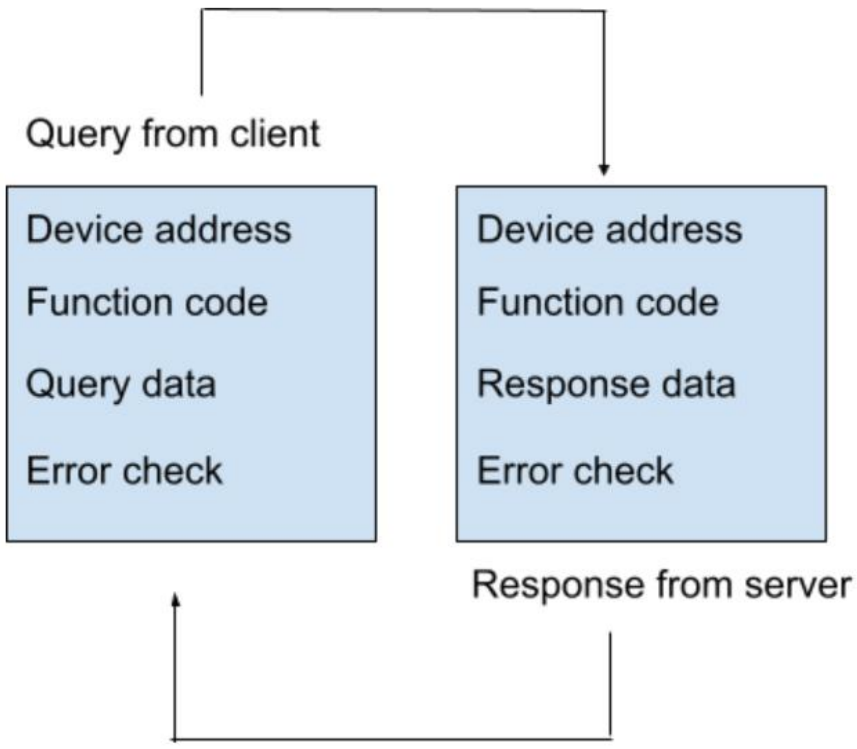


Figure 2: Client/server query response process

The function code tells the server device what action to perform when a request query is transmitted. The data, in this case, is used as additional information for the request. For example, function 03 will query the server to read the hold register and respond with content. The query data in this context are the memory addresses from where to read. (MODICON, Inc., Industrial Automation Systems, 1996)

In the server's response massage, the function code reflects the function code of the request query. If there is no error in the message, the response data field will contain the data requested by the master/client. If an error occurs, the function code is changed to indicate an error occurred, and the response data contains a code that describes the data. (MODICON, Inc., Industrial Automation Systems, 1996)

4.4.1 Modbus RTU

Field devices can use Modbus RTU to communicate over a serial network. The messages protocol for the Modbus RTU is like what the Modbus protocol describes. When implementing the Modbus RTU protocol, the serial connection only has (2^8)-1 or 255 different devices it can communicate with in the address field. CRC checking is applied to the whole message to detect errors. Figure 3 displays the OSI level 7 message data used in the Modbus RTU protocol.

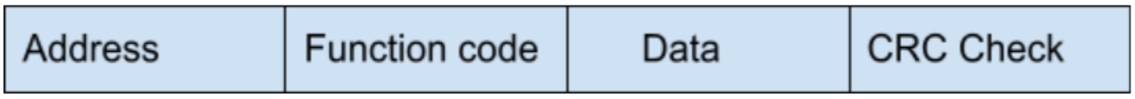


Figure 3: Modbus RTU Message header

4.4.2 Modbus TCP/IP

Like the Modbus RTU, there is Modbus TCP/IP. Modbus TCP/IP protocol also provides the resources for two devices to communicate over a local or wide area network. Compared to Modbus RTU, the TCP/IP version can use ethernet by TCP/IP (Acromag, 2005). The datagram header also slightly differs from that of the Modbus RTU. The checksum field of the message header is replaced as the TCP/IP layers supply this. In addition, Modbus Application Protocol Header (MBAP) replaces the address field. Figure 4 illustrates the differences.

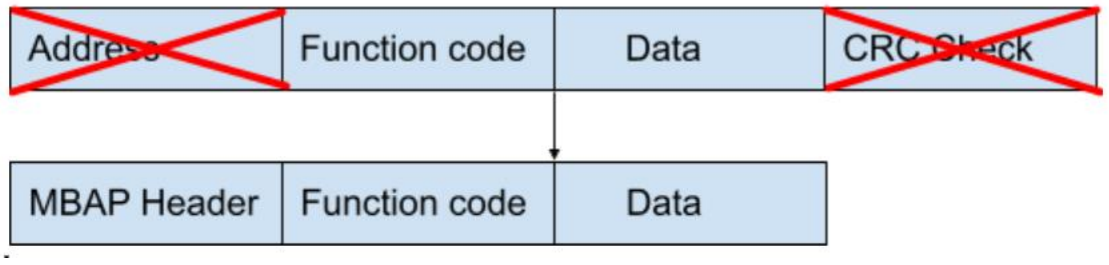


Figure 4: Difference between Modbus RTU and Modbus TCP/IP

4.5 Logs in the industrial control network

Logs in ICS are crucial for security and are part of what intrusion detection can analyze. As this thesis shows, the ICS network and its components can be sizable. The network has different segments, and each piece has various logging opportunities making it challenging to collect the logs as there is no central logging system. Field devices such as PLCs and RTUs do generally not have built-in functionality to keep track of logs, and to keep logs requires extra process power and storage (BDO AS, 2014). Even though it can be challenging to maintain logs for these field devices, it can be appropriate. There should be information about log-in attempts, connections, and the status of what ports are open. As for the non-field devices such as MTUs and HMI, there should be application logs, security logs, and system logs in place.

The application logs track what devices have one or more applications running when the application started, who opened the application, and actions run by the user.

Security logs are logs from the security aspects of the system. Here logs from firewalls, routers, switches, and IDS are kept with normalized times.

The system logs can get information about the system's operating status. The system log includes error and authentication messages when users run processes with elevated security status.

Other logging opportunities in the industrial network are Network/IPFIX, which shows the traffic flow in the network, offering meta-data about connections in the network, the ports connected to, the amount of data, and where the data is sent from and to (BDO AS, 2014). This data can be superior for an intrusion detection system using machine learning. It can difference between automated connections from components in the network based on time and irregular connections when there might be an adversary trying to make a replay attack.

The authentication logs show what users have accessed different services in the network, and the machine learning detection can use this to find outliers such as many failed log-in attempts. Authentication logs can also be beneficial for the use of anomaly detection. Table 3 Hovland, 2017 shows valuable data for ICS security.

Table 3: Potentially useful data from logs in ICS (Hvoland, 2017)

Log type	Data
application log	Date, Time Application name / ID User User actions
Security log	Date, Time Source, destination Port Protocol
System log	Date, Time Component / ID Process status Error message
Netflow/ IPFIX	Date, Time Source, destination Port Packet size Packet amount
Authentication log	Date, Time Service Source User Access rights

5 Intrusion Detection System (IDS)

Jim Anderson first proposed the idea of IDS in 1980 (Anderson, 1980). Intrusion is when an adversary takes actions aimed at compromising a system's confidentiality, integrity, and availability. The IDS prevents an advisory who tries to gain unauthorized access to the devices mentioned in chapter 2. IDS detects these advisories by providing well-established mechanisms, gathering data, and analyzing information from various areas of the host or network.

Ning & Jajodia's paper from 2003 concluded IDS can generally distinguish between an intruder’s behavior from a regular user’s traffic (Ning & Jajodia, 2003). Even though it is a powerful security tool, there are shortcomings, such as detecting unknown attack vectors.

Standard Intrusion detection function includes (i) monitoring and analyzing user, systems, and network activity. (ii) recognizing patterns of typical attacks (iii) analyzing patterns of abnormal activity. IDS works on assessing that intrusion activity are notably distinguishable from the benign activity and thus detectable. (Monowar, Bhattacharyya, & Kalita, 2014)

5.1 Host-based IDS and Network-based IDS

Intrusion detection methods can generally be classified based on their deployment. Deployment the two categories are 1) Host-based intrusion detection system (HIDS) and 2) Network-based intrusion detection system (NIDS).

HIDS monitors and analyses the internals of a computing system rather than its external interface. A HIDS might detect internal activity such as what program accesses what resource and illegitimate access attempts (Monowar, Bhattacharyya, & Kalita, 2014). SCADA system HIDS typically monitors system settings and configuration files, applications, and sensitive files (Knapp & Langill, 2014). HIDS resides in devices such as HMI or historian where such data is located. HIDS can only reside on a single device requiring multiple HIDS on multiple devices through the system.

On the other hand, NIDS deals with detecting intrusion using partially or whole network data traffic, causing fewer IDS to cover security across the entire system. The NIDS can alert about intrusion based on abnormal patterns in the network data generated by an adversary trying to get unauthorized access. The method NIDS operate is categorized based on the style of detection used. Signature-based detection searches the network traffic looking for known malicious behavior. Signatures are stored in a database and compared with network traffic (Joyothsna, Rama Prasad, & Munivara Prasad, 2011). These accelerate the detection of known attacks. However, as previously discussed, persistent attacks are on the rise, and adversaries are constantly tweaking known attacks or coming up with zero-day vulnerabilities, making these signature-based detection methods limited for attack detection.

The other category of NIDS is anomaly-based. Anomaly-based detection uses machine learning to either learn the underlying pattern of regular traffic or classify attack vectors that should generalize to new attack vectors. Researchers regard machine learning algorithms as efficient methods to improve the detection rate, reduce false alarm rate, and in the meantime, decrease computation and communication costs (M & Movahedi, 2015). However signature-based version is used in practice (Meng, 2011).

5.2 Overview of network anomaly detection

Anomaly detection attempts to find patterns in data that do not model the expected behavior in the network. Machine learning is a tool often used to accomplish this. Going back to the Stuxnet example, when the worm infected the SCADA system, it reprogramed the PLCs. Stuxnet would have been noticed with proper implementation of anomaly-based NIDS as it created unusual patterns in control network data.

Monowar, Bhattacharyya, & Kalita's 2014 paper has thoroughly gone through the aspects of anomaly-based NIDS. They start by classifying two broad categories of network anomalies. Performance-related anomalies and security-related anomalies. (Monowar, Bhattacharyya, & Kalita, 2014) The main difference between the two is that performance-based anomalies are benign in the case of cyber security and do not represent an adversary trying to invade the system. However, the NIDS will send an alert in both cases as the regular data traffic pattern has changed. For the SCADA system, the difference between these is significant as it will affect how to respond to the alarm. Reports such as (Ahmed, Parkash, & Zhou, 2020) look at different strategies to determine this, such as using data from both the network layer and limited log system, as this will give more certainty if there is an attack or a system fault.

Ideally, we want the NIDS to only respond to Security-related anomalies described in table 1. Monowar, Bhattacharyya, & Kalita divided security-related anomalies into three classes. 1) point, 2) contextual, and 3) collective anomalies (Monowar, Bhattacharyya, & Kalita, 2014). Table 4 describes these.

Table 4: Anomaly types and characteristics (Monowar, Bhattacharyya, & Kalita, 2014)

Types	Characteristics
1) Point anomaly	Point anomalies are instances of individual data found anomalous concerning the rest of the data.
2) Contextual anomaly	Contextual anomalies are data instances found abnormal in a specific context. The structure in the dataset induces context.
3) Collective anomaly	A collective anomaly is a collection of related data found anomalous concerning the entire dataset. The collection of events is an anomaly, but the individual events are not anomalies when they occur alone in the sequence.

5.3 The architecture of an anomaly NIDS

Monowar, Bhattacharyya, & Kalita 2014 go through what they have found generic in the architecture. Figure 5 depicts this generic architecture. An abstract view of how these are structured will help understand where the machine learning algorithms come into play.

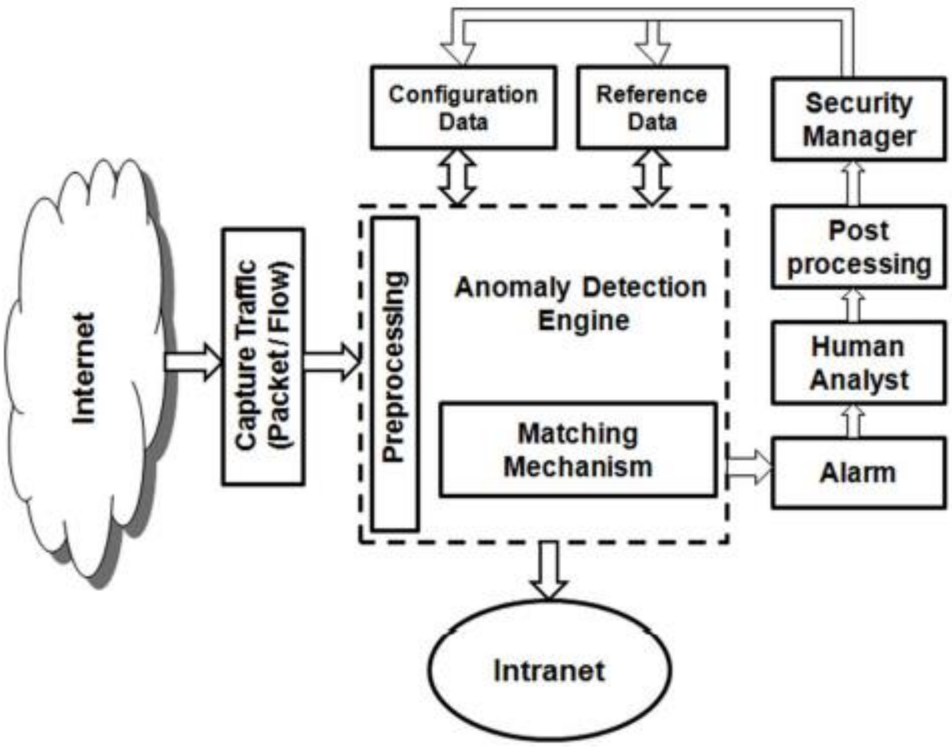


Figure 5: Generic architecture of NIDS (Monowar, Bhattacharyya, & Kalita, 2014)

These components are:

1) *Anomaly detection engine:* This is the core of the NIDS system. In the anomaly detection engine, the system tries to classify the into intrusions or not. Some machine NIDS even go as far as trying to classify the different types of attacks. The system must preprocess the data for a machine learning algorithm to understand the network data. The preprocessing step also helps remove biases and normalize the data, so network packet values with a high number do not significantly affect the machine learning algorithm. Monowar, Bhattacharyya, & Kalita also includes a matching mechanism. Signature-based NIDS uses this step to detect known attack vectors.
2) *Reference data:* The reference data stores the expected behavior of the system or known intrusions. Possible types of reference data in a generic ANIDS architecture are *signature, rule, and signature*.

3) *Configuration data:* The Configuration data contains the intermediate result of the detection engine. This component is mainly used in signature-based detection when partially created signatures resign.
4) *Alarm:* Component in the system responsible for generating alerts if there is a potential intrusion. Machine learning outcome triggers this alarm.
5) *Human analysis:* As IDS only creates an alarm by a suspected attack, there is needed a person responsible for analyzing, interoperating, and responding to the trigger alarm. The human analyst is also responsible for diagnosing the alarm data as a preprocessing step.
6) *Post-processing:* After an alarm is triggered, post-processing is a necessary step. The post-processing step diagnoses the event that provoked the warning, determining if a performance-based or a security-based anomaly caused the alarm.
7) *Traffic capture:* Tools used to capture the packet in the network traffic. These can be full packet captures or network flow statistics. Standard tools used are *Wireshark (packet capture), NFdump (flow capture), Nfsen (flow capture), and Cisco network flow (flow capture).*
8) *Security manager:* The security manager updates the stored signature and new signature.

5.4 Aspects of anomaly detection

Intrusion detection using machine learning generally is a classification or clustering problem. Classification and clustering are discussed in the machine learning part of the thesis. Chandola, Banerjee, & Kumar has, in their 2009 survey of anomaly detection, compacted different critical aspects of anomaly detection (Chandola, Banerjee, & Kumar, 2009). These aspects are 1) Types of input data, 2) Proximity measures, 3) Labels, 4) Classification based on available labels, 5) Feature identification, and 6) Reporting.

1) *Types of input data:* Before applying the input data from the network traffic, it is essential to describe the data types in the dataset attributes. These attributes can be binary, categorical, or numeric. In the case of network traffic, the data is often

multivariable, containing multiple attribute types. Each of these types requires different preprocessing to optimize machine learning results.

2) *Proximity measures:* Proximity measures are necessary for solving classification and clustering problems. Using a different measure of the error in classification or distance between the data can change the outcome of the machine learning result.
3) *Labels:* In the machine learning landscape, labels represent the class to which the data sample belongs. In the case of network traffic, this can either be binary or multiclass. Binary classification labels network traffic as benign or anomalous—multiclass classification labels traffic data into multiple attacks.
4) *Classification based on available labels:* Considering the extent of available labels for the data. Anomaly detection can work in 3 modes, supervised, unsupervised and semi-supervised. The Machine learning chapter of the thesis explains these further.
5) *Feature identification:* What features from the network data can significantly impact the learner's result. The learning process can reduce computational complexity by removing irrelevant features, removing information redundancy, increasing performance, facilitating data understanding, and improving generalization. (Monowar, Bhattacharyya, & Kalita, 2014)
6) *Reporting:* The learning algorithm has multiple output types for the input data These can be a score representing the output compared to a threshold representing the anomaly rate. In the case of multiclass attack classification, a vector contains the attack class's likelihood. Or, in the case of anomaly detection, a binary label, either abnormal or normal.

5.5 The challenges of Anomaly detection in networks

Even though NIDS and machine learning have been extensively researched, signature-based NIDS is deployed in practice. There are multiple reasons why this is the case, and known shortcomings of anomaly detection in NIDS include: a high cost of error, lack of training data, a semantic gap between results and their operational interpretation, enormous variability in input data, and fundamental difficulties for conduction sound evaluation (Sommer & Paxson, 2010). A high error cost is actual for SCADA systems as for traditional networks, not detecting an attack renders the system useless. On the other hand, if the system classifies many anomalies correctly but does this because it triggers many alarms and falsely sends alarms for benign data. A human analyst would have to look through all the alarms, which would be expensive. In other fields where machine learning accelerates, such as recommendation systems, a 90 percent accuracy is more than enough. It does not lead to a fatal error if the algorithm predicts wrong. In intrusion detection, this is a horrible result. Since the Sommer & Paxson 2020 report, there is done considerable development in the benchmark dataset used for traditional networks. However, the lack of training data for SCADA networks is a genuine concern.

6 Machine learning

When talking about machine learning, it is often in the case of AI. AI is a term that has been much used and popularized in the mainstream by popular films such as “terminator” and “the matrix,” where artificial general intelligence has become more intelligent than humans. A subcategory of this general intelligence is machine learning. Machine learning is a subcategory of AI when statistical models are used to automatically “learn” or improve when performing a task. Machine learning is nothing new. The term has been around since 1940 when Walter Pitts and Warren McCulloch developed the early stages of what is today called a neural network. Taking inspiration from the fundamental units in the brain called neurons, they created the first application of supervised learning. However, it was not until the 1980s that back-propagation, increased processing power, and data availability accelerated the landscape of machine learning. Today machine learning is generally classified based on the learning methods used. These methods include 1) Supervised- 2) unsupervised- and 3) reinforcement learning.

6.1 Supervised learning

Supervised learning learns by using labeled data representing binary or multiple classes. The learning algorithm trains to classify new data based on the labels. In NIDS, the labels for each instance of the network data can be binary such as normal and abnormal, or multiclass such as different attack categories shown in table 1. Other than classification, supervised learning algorithms may use regression trying to predict some numerical value. Supervised learning generally has better results than unsupervised and reinforcement learning counterparts. A drawback of supervised learning algorithms is the requirement of labeled data, which is tedious and often requires expertise, especially if the dataset is not ubiquitous (e.g., pictures of cats and dogs), such as network data. In the case of intruder detection, there is another major issue: there are far fewer instances of intrusion data than regular traffic. (Mantere, Sailio, & Noponen, 2015)

6.2 Unsupervised learning

Opposite to supervised learning, there is unsupervised learning. There are no labeled samples in unsupervised learning. Unsupervised learning solves tasks such as clustering, dimensionality reduction, visualization, association rule learning, and anomaly detection (Gèron, 2019). Figure 6 displays unsupervised learning does anomaly detection.

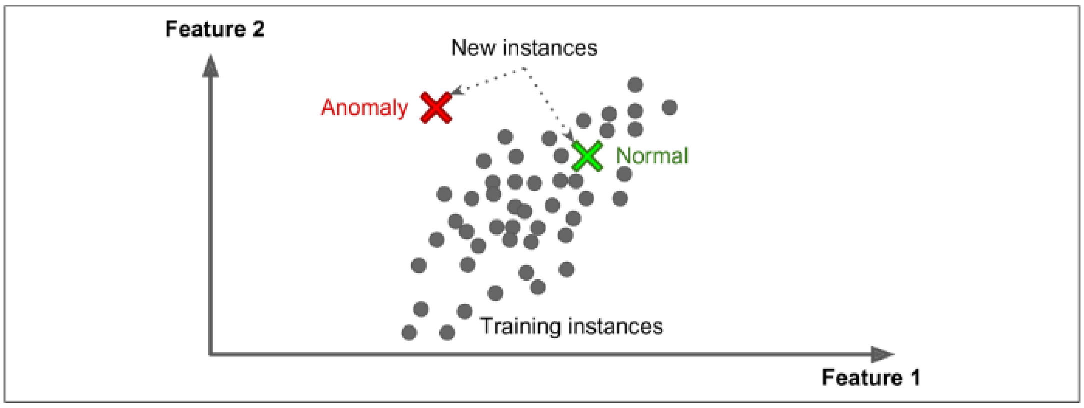


Figure 6: Unsupervised learning for anomaly detection (Gèron, 2019)

6.3 Semi-supervised learning

Semi-supervised learning also acknowledges the flaw of requiring fully labeled datasets. However, instead of not having any instances labeled, such as in unsupervised learning, generally, a small portion of the data is labeled. The rest of the dataset is labeled based on these. Often by using a combination of super- and unsupervised learning methods, the semi-supervised. An example of semi-supervised learning is adding photos to a photo gallery on the phone. When naming a person on some pictures, the algorithm labels the rest (Gèron, 2019). Figure 7 illustrates this.

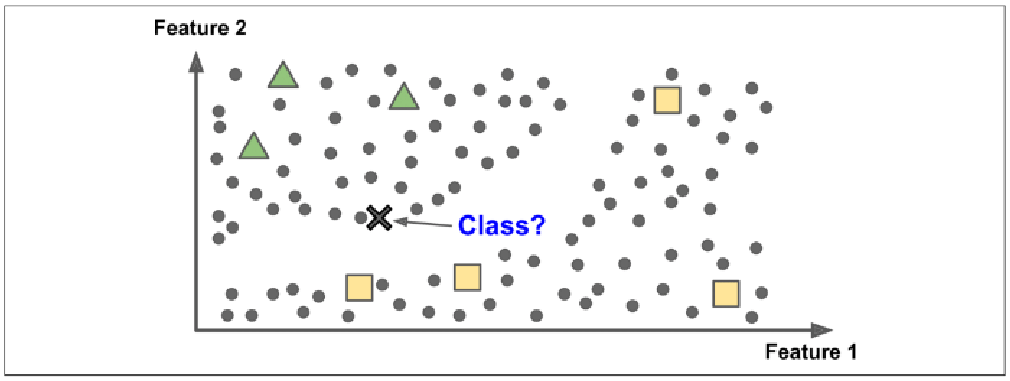


Figure 7: Self-supervised learning example showing some labeled classes (Gèron, 2019)

6.4 Reinforcement learning

Lastly, there is reinforcement learning. Reinforming learning is very different from the learning methods above. As was the case for unsupervised learning, the data is not labeled. Instead, the learning system, called an agent, selects and performs actions according to what gives the most reward (displayed in figure 8). Reinforcement learning has had astonishing results and was used to create deep blue beating human former chess champion Garry Kasyanov (Campbell, Hoane, & Feng-hsiung, 2002). Researchers have applied reinforcement learning has also to the task of NIDS. Such as (Sewak, Sahay, & Rathore, 2022).

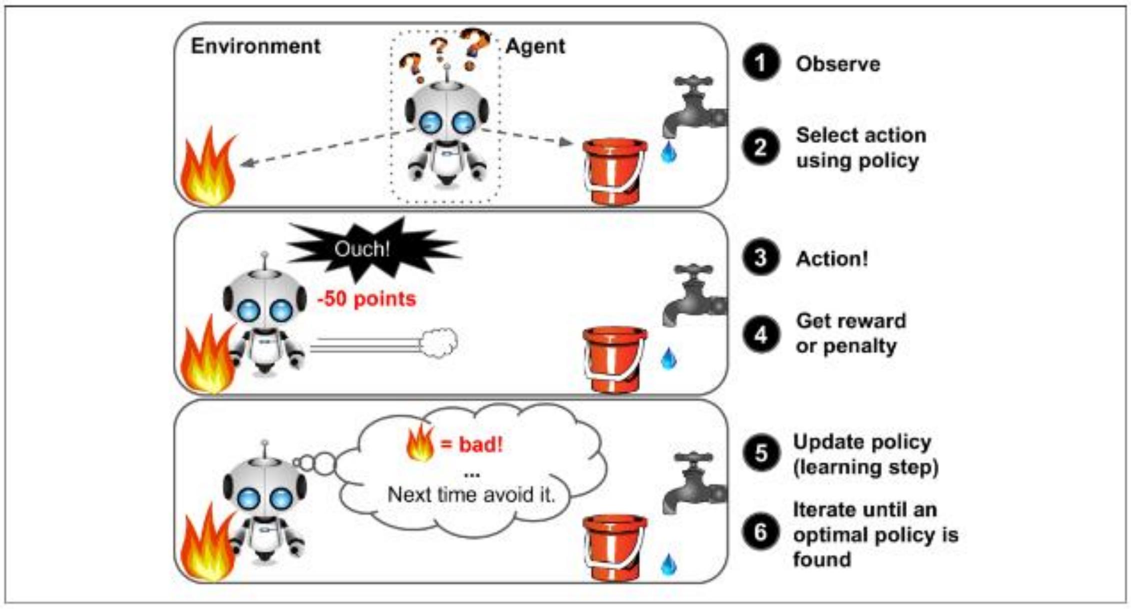

Environment
Agent
1 Observe
2 Select action using policy
Ouch!
-50 points
3 Action!
4 Get reward or penalty
= bad!
...
Next time avoid it.
5 Update policy (learning step)
6 Iterate until an optimal policy is found

Figure 8: Basics of reinforcement learning (Gèron, 2019)

6.5 Deep learning

As stated above, researchers have researched machine learning for quite some time. In more recent times, branching out from machine learning is deep learning. The difference between machine- and deep learning is that deep learning tries to mimic further how the brain works. Recent deep learning strategies have significantly improved performance in computer vision, natural language processing, and other predictive tasks (Fan, Cong, & Zhong, 2021). These techniques are generally more efficient than the ML due to their deep structure and ability to learn the essential features from the dataset on its own to generate an output (Ahmed, Parkash, & Zhou, 2020). However, they usually take a longer time to train because of their complexity.

6.6 Machine learning algorithms used for anomaly detection engine

Researchers have applied numerous unique machine learning algorithms for the NIDS detection engine. The function of these algorithms is to detect outliers among the input data by some threshold that determines what “normal” input is. For this task, all the above learning methods approaches are researched. However, supervised learning generally has a better detection rate (Joshi, 2017).

The 2020 survey by Ahmed, Parkash, & Zhou provided an overview of the anomaly detection engine's most commonly used machine- and deep- learning algorithms.

For the machine learning models, they include decision trees (DT) such as random forest and K-nearest neighbor (k-NN), support vector machine (SVM), k-means clustering ensemble learning (EL), and artificial neural networks (ANN). (Ahmed, Parkash, & Zhou, 2020)

Whereas for deep learning, the most used algorithms include Recurrent neural networks (RNN) with types such as Long short-term memory (LSTM), deep neural network (DNN), Convolutional neural network (CNN), and Auto encoders (AE) with types such as stacked, sparse and variational AE.

In SCADA system (Alimi, Ouahada, Abu-Mahfouz, Rimer, & Alimi, 2021) mentions supervised learning algorithms such as SVM, k-NN, DT, and Random Forest.

6.7 Machine learning algorithms

6.7.1 Support vector machine (SVM)

An SVM is a powerful unsupervised machine learning model that can perform linear, non-linear, regression, and outlier detection—all of which are used to detect intrusion in network data (Gèron, 2019).

6.7.2 Decision trees (DT)

Decision trees are a supervised learning method. Like SVM, decision trees can perform various tasks such as classification, regression, or output of multiple values. A Decision tree classifies the samples through a sequence of decisions. Each branch of the sequential model represents the outcome of a test. The previous choices help decide the following in the series.

6.7.3 Ensemble learning

Ensemble refers to a group of predictions. The essence of ensemble learning is that multiple predictions are better than one. If a large number of non-expert people take a multiple-choice exam and we compare the results of all the exams, the most chosen answer to each question is probably the right one. Comparing this to if a single expert answers the exam, the expert might slip up getting the wrong answer.

6.7.4 Random forest

Random forest uses ensemble learning in a supervised structure. By training a DT on different subsamples of the data and comparing the results, the model might get higher performance than if training a single DT classifier.

6.7.5 Isolation forest

Isolation forest works much the same way as Random forest by using DTs. The Isolation forest, however, is an unsupervised learning model. The Isolation forest can classify anomalies by calculating the path length to an observation in the DT.

6.7.6 K-nearest neighbors (k-NN)

K-nearest neighbors is a simple yet effective supervised learning algorithm. When classifying unknown data, the algorithm measures the distance from the labeled data giving the new data the same class as the k-nearest datapoints. K-NN can classify complex data with multiple inputs. Even with the algorithm's simplicity, there are many variants of the k-NN model as there are many ways used to calculate distance. Different distance measurements can be Makowski-, Manhattan- or L1 -, Euclidean- or L2, cosine- and Jaccard distance.

6.7.7 K-mean clustering

K-means clustering is a form of unsupervised clustering. This learner chooses a random starting point. The starting point is called the centroid. The k in k-mean refers to the number of centroid nodes. The distance to the centroid decides the class of the data. Next, the algorithm finds new centroids, the center of the clusters created, and groups the data by the distance to the new centroid. This process happens over multiple epochs until there is little to no change centroid.

6.7.8 Artificial Neural networks (ANN)

An artificial neural network is a supervised learning model built to mimic the human brain (in a trivial sense). By stacking multiple layers of neurons/perceptrons, the ANN manages to classify data in a non-linear manner. Making the ANN is beneficial in more complex data that is not linearly separable (such as network data).

NN has three layers: 1) Input layer, 2) hidden layer, and 3) hidden layer.
1) The NN consists of a single input layer. This layer has the exact dimensions of the input data.
2) The hidden layer(s) singular or plural depending on the task. The dimension of the hidden layer(s) or the number of nodes is decided based on the assignment and often tuned to get the optimal result.
3) When the data has sequentially traversed the input and hidden layer(s), it is sent to the output layer/node(s). The desired result decides the dimensions of the output layer.

ANNs use back-propagating to train. The model tunes the different weights between the nodes, gradually getting a better result on the labeled training data.

6.8 Deep learning algorithms

Deep learning describes more complex ML models. We also split DL models into supervised, unsupervised, and reinforcement learning models. For the sake of using multiple learning types on the datasets. We selected to apply a supervised and unsupervised model. Those depicted by (Ahmed, Parkash, & Zhou, 2020) for intrusion detection in the supervised category is Recurrent Neural Networks (RNN) with types such as Long short-term memory (LSTM). The unsupervised approach includes Autoencoders (AE).

6.8.1 Recurrent Neural Networks (RNNs)

RNN accelerates in sequence processing, with tasks such as natural language processing and language translation. In essence, RNNs are the same as ANNs. The difference, however, is that RNNs consist of multiple layers of ANN. By stacking numerous ANNs, the model maintains data from previous iterations.

6.8.2 Long short-term memory (LSTM)

As recurrent neural network seems like a good alternative for classification on the sequential dataset, RNNs have some problems that can be solved using another model. However, RNN has some known issues, namely vanishing gradient. LSTM is a form of RNN where the neural network has a feedback loop. The model is built differently compared to traditional RNN. It consists of a forget-, input, and output gate. With the help of these gates, the LSTM model can keep only the relevant information over multiple longer timeframes.

The LSTM model remembers previous states making it useful for work requiring memory and state awareness (Shrestga & Mahmood, 2019). LSTM feeds data multiple times through the network, making them favorable for recognizing patterns. In addition, LSTM only keeps the most helpful information and remembers information from data processed earlier in the sequence.

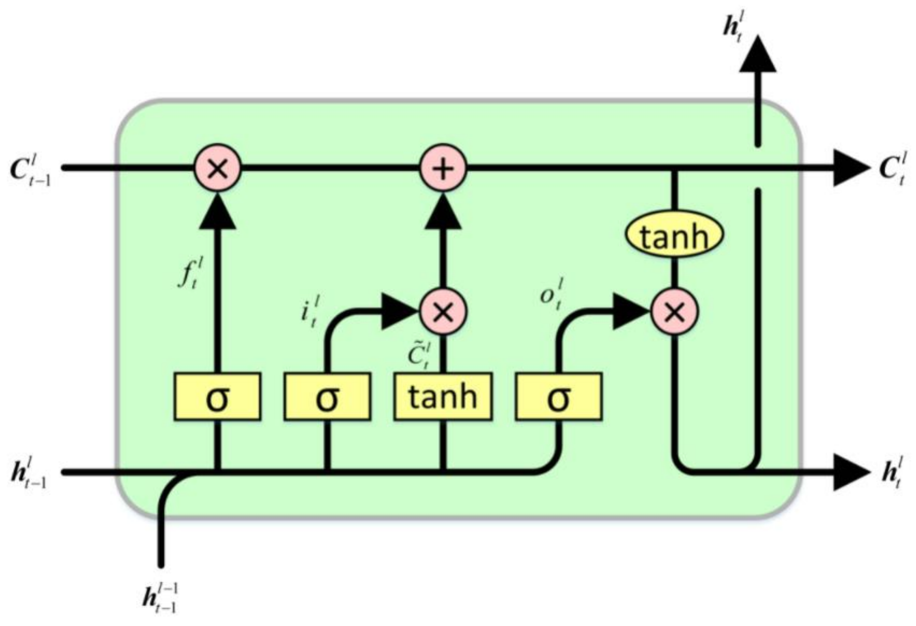


Figure 9: LSTM model (Varsamopoulos & Bertels, 2018)

As LSTM is particularly good at detecting patterns in data sequences, it is efficient at detecting attacks such as distributed denial of service (DDOS) and man-in-the-middle (MITM) attacks (Gao, et al., 2019). Typical machine learning algorithms such as logistic regression or SVM cannot detect inter-packet patterns (Gao, et al., 2019). As they are an improved version of RNNs and with their power to detect patterns, LSTM will be selected as a supervised deep learning model.

6.8.3 Autoencoder

In anomaly detection in industrial networks, there is a high cost of data that is well balanced, containing an equal amount of bough animalities and regular traffic. An approach to work around this problem is to use un- or semi-supervised learning.

Autoencoders, are models that learn complex non-linear relationships between the data points. The autoencoder model uses multiple neural network layers as an encoder and decoder block. In the encoder part, the network tries to recreate the data trained on by training and fine tuning with back-propagation. On the other hand, the decoder attempts to discriminate between the data created by the encoder and the original data. The model compresses the high dimensional data X into a lower dimension Z before being recreated into the original data X’. With fewer nodes in the lower representation of the data, the model must carefully choose the most critical aspects of the input and ignore the noise. Common uses for autoencoder can be compression, recommendation systems, outlier detection(anomaly), and image generation (Gèron, 2019). We use the Autoencoder DL model as an unsupervised model on the datasets.

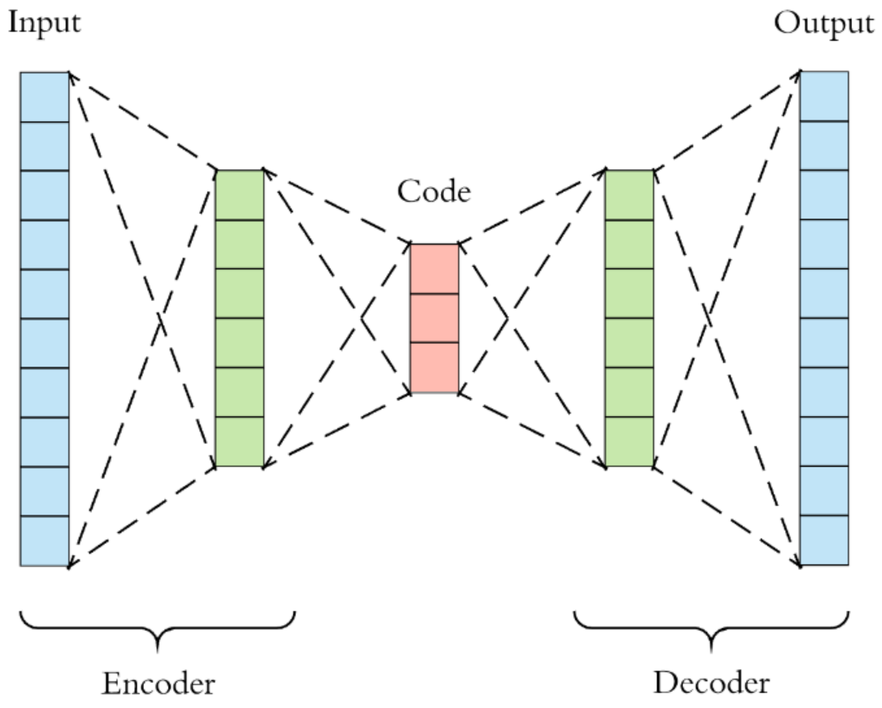


Figure 10: AE model (Dertat, 2017)

6.8.3.1 Autoencoder for anomaly detection

In an autoencoder for anomaly detection training, only normal traffic is input for the encoder. After training, the lower dimensional layer (red layer) will learn the latent representation of the normal traffic. The decoder uses the latent representation to reconstruct the original input data. When there is an anomaly, the decoder will have difficulty reconstructing the original data, making the reconstruction error high. The inputs can be flagged as anomalies using a threshold.

6.9 Evaluating machine learning algorithms

When evaluating machine learning algorithms, accuracy, the amount of correctly classified labels, is commonly used as a performance measurement. In intrusion detection, this might not be optimal. As mentioned previously, normal traffic is generally more common than malicious traffic. Considering this, if a dataset contains 99% regular traffic and only 1% malicious traffic, an accuracy score of 99% might seem high, but the model has essentially only classified data as normal.

Therefore, other measurements are used to calculate the performance of the system: precision, recall, and F1-score.

Precision, recall, and F1-score are calculated based on the different outcomes of a machine learning model for network intrusion detection. The possible results can be:

- True positive (TP): Intrusion sample classified as an intrusion.
- True negative (TN): Normal sample classified as normal.
- False positive (FP): Normal sample classified as an intrusion.
- False negative (FN): Intrusion sample classified as normal.

Accuracy is useful in the case of evenly distributed intrusion and normal samples. Accuracy can be calculated by:

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

Equation 1: Accuracy

Precision indicates how precise the machine learning model is. And is calculated by the formula:

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

Equation 2: Precision

However, a trivial way to have perfect precision is making a single positive prediction ensuring this is correct. Considering this Recall, also called sensitivity, true positive rate, or in the case of intrusion detection, attack detection rate, is combined with precision. The recall is the rate of positive instances correctly detected by the classifier. The recall is calculated by:

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

Equation 3: Recall

These measurements are often combined to evaluate machine learning algorithms as they can directly affect each other. The harmonic score between precision and recall is called the F-1 score and is calculated by:

$$F1\_Score = \frac{2 * Precision * Recall}{Precision + Recall}$$

Equation 4: F1-Score

7 Method

The first part of the thesis went through the fundamentals of the experiment. The method chapter will apply a machine learning algorithm to an industrial and a traditional network. For the results of the models to be comparable, the models will be kept consistent on bough datasets—a structured approach to applying these models is required. Gèron's 2019 book "hands-on machine learning" suggests a structured checklist for going through machine learning projects (Gèron, 2019).

7.1 Machine learning checklist

Following the process for how to undergo a machine learning project, Gèron 2019 checklist has many appropriate proposals for how this machine learning project should unfold. Gèrons checklist items that are useful for this thesis include:

1) Framing the problem and looking at the big picture
2) Get the data
3) Explore the data to gain insight
4) Prepare the data to better expose the underlying data patterns to machine learning algorithms
5) Explore many different solutions and shortlist the best ones
6) Fine-tune your models and shortlist the best ones

Implementation of items [1-4] is in the method chapter. Implementation of items [5-6] is in the results chapter.

7.2 Framing the problem and looking at the big picture

Starting by framing the problem, Gèron suggests defining the objective in business terms. The aim of these machine learning algorithms is as a NIDS detection engine. The learning algorithm should be able to signal intrusions. The solution should be implemented on the network, analyzing network traffic captured through tools described in the NIDS architecture. Current solutions to the problem are the use of signature-based network intrusion detection.

There are multiple ways to frame this problem. Network intrusion detection is generally solved using supervised learning algorithms with classified intrusions or using an unsupervised approach to distinguish between normal and abnormal data. Often in classification problems, accuracy is used as the primary performance measurement. In the case of intrusion detection, this can give misleading results. Intrusions are often rare, meaning data would probably contain low amounts of positive samples. Performance measurements used are recall, precision, and f1-score to analyze the results. Emphasized is F-1-score as it is a combination between recall and precision.

The performance should be high enough for an analyst to find the system useful. If the algorithm produces classifies many intrusions that are normal behavior, the system will be rendered useless and lead to more work for an analyst.

When framing the problem, Gèron, 2019 also suggests comparing the task at hand with issues in other fields. A similar situation to that of intrusion detection is fraud detection. In fraud detection, many banks have samples of normal behavior for a user while trying to classify strange behavior that can suggest fraud or stolen credit cards. Methods and studies from fraud detection can be helpful and possibly reused in the case of network intrusion detection. Trying to solve this problem manually would be to use a tool such as Snort writhing rules alerting about known intrusion traffic.

7.3 Get the data

Required for this experiment is traffic from traditional and ICS networks. Table 5 list a collection of open-source datasets. However, most of these datasets only provide logfiles, including data described in the log chapter. Only “SWAT,” “Water tank,” “Electra,” and “ICS-PCAP” contain network data from different protocols. This experiment has a considerable advantage if the labeled dataset leaves out the “ICS-PCAP” dataset. The remaining datasets all capture distinct protocols from the network. As mentioned in the network protocols chapter, the Modbus protocol is the most widely implemented application layer protocol in ICS. “Water tank” and “Electra” datasets capture this protocol. In the Electra dataset, however, the Modbus TCP/IP protocols are implemented, which is a more modern version than Serial Modbus. The Electra dataset also contains the widest variety of attacks. The experiment will use Electra as a control network dataset.

Compared to ICS network traffic, a wide array of benchmark traditional network traffic datasets exists. Table 6 lists these benchmark datasets. The work of (Ahmed, Parkash, & Zhou, 2020) explains that the most used of these datasets is the NSL-KDD, where 60% of newer studies on traditional network anomaly-based NIDS use this for evaluating machine learning models. However, they mention that the dataset is old. We outway this as the network attacks are somewhat similar to those in the SCADA network, making them more comparable. We describe these attacks in the attack implementation below.

Table 5: ICS datasets

Datasets	**Provides**	**Protocols**	**Attacks**
SWAT (Mathur & Tippenhauer, 2016)	Packets Sensors/ Actuators logs	CIP EtherNet/IP	False Data Injection
WADI (Water Distribution (Wadi), 2015)	Sensor/ Actuators logs	-	False Data Injection
EPIC (Adepu, Kandasamy, & Mathur, 2019)	Packets Sensors/ Actuator state	-	False Data Injection
Power system (Pan, Morris, & Adhikari, 2015)	Logs Sensors/ Actuator state	-	False data Injection
Gas Pipeline (Beaver, Borges-Hink, & Buckner, 2013)	Precomputed features from RTU Telemetry	-	False data Injection
Water Tank (Morris, Srivastava, Reaves, & Gao, 2011)	Precomputed Features	Serial Modbus DNP3	False Data Injection DoS Reconnaissance
Electra (Gòmez, et al., 2019)	Precomputed network traffic features	Modbus TCP/IP S7Comm	False Data Injection Replay Reconnaissance
ICS-pcap (Smith, 2016)	Collection of PCAPs for ICS/SCADA utilities and protocols.	DNP3 MODBUS S7 Comm and more	Normal traffic

Table 6: Traditional network datasets

Datasets	**Provides**	**Protocols**	**Attacks**
Darpa98 (Darpa98, 1998)	Packets	TCP/IP	DoS R2L U2R Probe
KDDCup99 (KDD99, u.d.)	Precomputed network features	TCP/IP	DoS R2L U2L Probe
NSL-KDD (NSL-KDD, 2015)	Precomputed network features	TCP/IP	DoS R2L U2L Probe
CTU-13	Packets without payload flow	TCP/IP	Botnet
CICIDS (CICIDS, 2011)	Packet flows	TCP/IP	Brute Force DoS and DDoS Hearthbleed Web attack Infiltration Botnet
NGIDS-DS (NGIDS-DS, 2016)	groundtruth.csv CSV files of host log, Pcap of the network packets	TCP/IP	DoS Worms Reconnaissance Shellcode Backdoor
UNSW-NB15 (UNSW-NB15, 2021)	Pcap files, BRO files, Argus Files, CSV files, and the reports	TCP/IP	Fuzzers Analysis Backdoors DoS Exploits Generic Reconnaissance Shellcode Worms

7.4 Exploring the dataset

This part (Gèron, 2019) suggests steps such as studying each attribute and its characteristics. They can be names, type, task usefulness, noise, and kind of noise and distribution. He also suggests visualizing the data and studying the correlations and promising transformations. Lastly, he suggests identifying extra data that would be useful. Additional features can be helpful for future work for control networks.

7.4.1 NSL-KDD dataset

The NSL-KDD dataset is not the first dataset of its kind. The dataset stems from the KDD’99, where KDD stands for knowledge and data mining competition. The competitors would create an intrusion detection system distinguishing bad and good connections in the competition. As a result of the competition, internet traffic was collected and put together to form KDD’99. NSL-KDD is a further improvement of the KDD’99, which has been clean up and revised by the University of New Brunswick (Saporito, 2019). Compared to the KDD’99, the NSL-KDD 1) does not include redundant records in the training set, 2) No duplicate records in the test sets, 3) The number of selected records from each difficulty level group is inversely proportional to the percentage of records in the original KDD data set, 4) The number of records in the train and test sets are reasonable (NSL-KDD, 2015).

7.4.2 NSL-KDD Attack implementation

The NSL-KDD dataset contains 37 different attack types (Protić, 2018). These 37 attacks are classified into four categories which are 1) Denial of Service (DoS). 2) Probe 3) User to Root (U2R). 4) Remote to Local (R2L). These attack types slightly differ from those described in table 1 as traditional networks have different attacks.

1) DoS includes attacks where the adversary increases the network traffic to limit the availability of a service
2) Probe attacks include attacks where the adversary collects information about the network and hosts to discover vulnerabilities. The probe attack is comparable with the analysis and reconnaissance attacks in table 1.
3) U2R are attacks where the adversary escalated privileges (root) in the network.

4) R2L are attacks where the adversary tries to gain access to the system by getting the account of a typical user.

Table 5 displays the distribution of the attack categories.

Table 5: Distribution of data in NSL-KDD (Saporito, 2019)

	Total	Normal	DoS	Probe	U2R	R2L
Train	125973	67343 (53%)	45927 (37%)	11656 (9.11%)	52 (0.04%)	995 (0.85%)
Test	22544	9711 (43%)	584 (33%)	2421 (11%)	200 (0.9%)	2654 (12.1%)

7.4.3 NSL-KDD Dataset attributes

Table 6 lists the data attributes of NSL-KDD. A detailed description of the features is in works such as (Dhanabal & Shantharajah, 2015).

Of importance for the experiment is attributed classification. There are four classes of data types for attributes, namely 1) categorical, 2) binary, 3) discrete, and 4) continuous.

1) Categorical features are often of type: object. The data is usually a fixed number of classes. The Categorical features from table 6 are: [2, 3, 4, 42]
2) Binary features represent features that are either 1 or 0. Binary features in table 6 are: [7, 12, 14, 15, 21, 22]
3) Discrete features have a fixed range of possible values are: [8, 9, 15, 23-41]
4) Continuous features have an infinite range of possible values: [1, 5, 6, 10, 11, 13, 16-20]

Table 6: Attributes of NSL-KDD

Nr	Name	Type	Nr	Name	Type
1	‘duration’	Int	22	‘is_guest_login’	Int
2	‘protocol_type’	Obj	23	‘count’	Int
3	‘service’	Obj	24	‘src_count’	Int
4	‘flag’	Obj	25	‘serror_rate’	Float
5	‘src_bytes’	int	26	‘svr_serror_rate’	Float
6	‘dst_bytes’	Int	27	‘rerror_rate’	Float
7	‘land’	Int	28	‘srv_rerror_rate’	Float
8	‘wrong_fragment’	Int	29	‘same_srv_rate’	Float
9	‘urgent’	Int	30	‘diff_srv_rate’	Float
10	‘hot’	Int	31	‘srv_diff_host_rate’	Float
11	‘num_failed_logins’	Int	32	‘dst_host_count’	Int
12	‘logged_in’	Int	33	‘dst_host_svr_count’	Int
13	‘num_compromised’	Int	34	‘dst_host_svr_count’	Float
14	‘root_shell’	Int	35	‘dst_host_diff_srv_rate	Float
15	‘su_attempts’	Int	36	‘dst_host_same_src_port_rate’	Float
16	‘num_root’	Int	37	‘dst_host_src_diff_host_rate’	Float
17	‘num_file_creations’	Int	38	‘dst_host_serror_rate’	Float
18	‘num_shells’	Int	39	‘dst_host_srv_serror_rate’	Float
19	‘num_access_files	Int	40	‘dst_host_rerror_rate’	Float
20	‘num_outbound_cmd’	Int	41	‘dst_host_rerror_rate’	Float
21	‘is_host_login’	Int	42	‘class’	Obj

7.4.4 NSL-KDD object description

Traditional networks use many services, status flags, and protocol types. The numerous object types will lead to significantly more input dimensions for the machine learning algorithms as these have to be preprocessed for the learner to understand. In table 7, there is an overview of the categorical attributes in the NSL-KDD dataset.

Table 7: Categorical attributes of NSL-KDD

```
http            40338
private         21853
domain_u         9043
smtp             7313
ftp_data         6860
                ...
tftp_u              3
harvest             2
aol                 2
http_8001           2
http_2784           1
Name: 'service', Length: 70, dtype: object
```

```
SF          74945
S0          34851
REJ         11233
RSTR         2421
RSTO         1562
S1            365
SH            271
S2            127
RSTOS0        103
S3             49
OTH            46
Name: 'flag', dtype: object
```

```
tcp      102689
udp       14993
icmp       8291
Name: 'protocol_type', dtype: object
```

```
normal     67343
anomaly    58630
Name: 'class', dtype: object
```

7.4.5 NSL-KDD Test set

As mentioned in the attack implementation, there are 37 different attacks implemented. Implemented in the training set are 21 of these. However, the test sets have implemented all types (Protić, 2018). As APT and zero-day vulnerabilities are the main reason for machine learning-based NIDS, this train test structure is highly recommended but can affect the learner's score. Because of this, one can be sure that the learner generalizes well.

7.4.6 Electra dataset

Network traffic of an electric traction substation running in normal conditions and under attack generates the Electra dataset. (Perales Gòmez, et al., 2019) created the dataset in realistic scenarios with standard industrial devices such as PLCs and a SCADA system communicating with well-known industrial protocols such as S7Comm and Modbus (Perales Gòmez, et al., 2019). The data in the dataset is split into two files comma-separated value (CSV) files. One subset for the Modbus protocol and one for the S7Comm protocol.

To create the dataset, the authors used an Electric traction substation, which purpose is to convert electric power from the form provided by the electrical power industry to the correct voltage, current, and frequency to support railways/trams. The testbed comprises one master PLC, four slave PLCs, a SCADA system, a switch for the interconnection of different devices, and a firewall. Communication protocols used are Modbus TCP, OPC (communicate), and S7Comm. Figure 11 displays the topology for the network. The figure shows a simple ring topology connecting the Modbus PLC devices.

For the Modbus subset, 94.8% of the data is the normal flow of the system, which is kind of high and essential to consider when applying deep learning to anomaly detection.

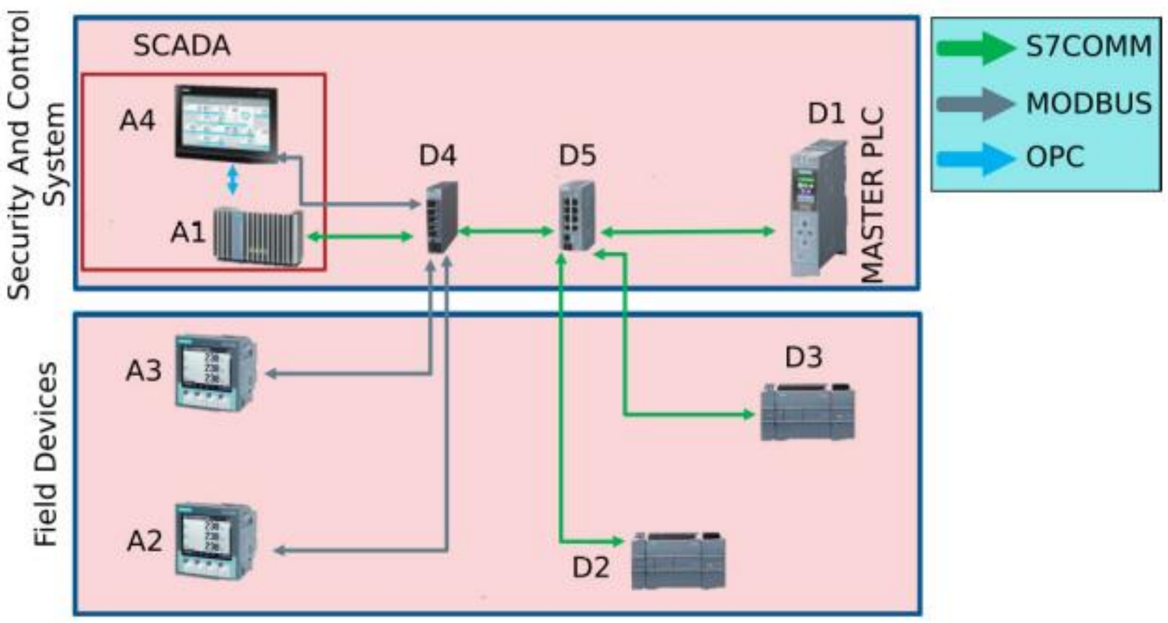


Figure 11: Electra network topology (Perales Gòmez, et al., 2019)

7.4.7 Electra attack implementation

There are three categories of attacks 1) Reconnaissance attacks, 2) False data injection attacks, and 3) Replay attacks. Table 8 lists the attacks in the dataset.

There are two types of attacks 1) create packets to perform spurious writes or read in valid memory of a PLC 2) modify existing packets to alter the data returned by the server PLC (Gòmez, et al., 2019).

Table 8: Electra Modbus attack types (Gòmez, et al., 2019)

Category	**Attack**	**Type**
Reconnaissance	Function codes recognition	Packet creation
False data injection	Response Modification	Packet modification
	Forced error in response	Packet modification
	Command Modification	Packet modification
	Read data	Packet creation
	Write data	Packet creation
Replay	Replay valid packets	Packet creation

A new node planted in the network creates the attacks. It was used as a MitM node and configured to implement the false data injection attacks by poisoning the network device's Address Resolution Protocol (ARP). This way, the new node has access to all the messages exchanged. The replay and reconnaissance attacks were implemented in python using the standard library socket class. These attack types are relevant and well selected as they are on different stages of the cyber kill chain.

7.4.8 Electra dataset attributes

Table 9 displays the attributes in the Electra Modbus dataset. When looking at the dataset's features, we can see eleven values, including the label. The authors have selected to include only the attributes in the Modbus protocol in addition to MAC/IP addresses, removing other header fields of Ethernet, TCP, and IP Protocols. (Gòmez, et al., 2019) From figure 11, we see that 4 of the values are objects which must be preprocessed for machine and deep learning to work. From the previous figure 11, we see four devices communicating by following the grey lines. The different data types for the attributes are:

1) Categorical features are: [2-4]
2) The binary feature is: [6]
3) Discrete features are: [7-10]
4) Continuous features are: [1]

Table 9: Attributes of Electra Modbus

Nr	Feature name	Description	Data Type
1	Time	Timestamp	Int
2	Smac	Source mac address	Object
3	Dmac	Destination mac address	Object
4	Sip	Source IP address	Object
5	Dip	Destination IP address	Object
6	Request	Indicates whether the packet is a request	Int
7	Fc	Function code	Ing
8	Error	Error code	Int
9	Address	Address to perform operation	Int
10	Data	Indicates data to send to client in case of read. In case of write indicates the data client sends to the server PLC	Int
11	Label	A label indicating attack type	Object

7.4.9 Addressing Electra Modbus duplicates

NSL-KDD does not include duplicate network traffic. For consistency, we removed duplicates from the Electra Modbus dataset. After removing the duplicate data samples, the dataset went from 16289277 records to 41429. This step is required because control processes repeat the same actions over time, and machine learning can observe the repetitive nature of the network traffic (Gòmez, et al., 2019). With the new subsample of the data, there is a [24409, 17020] Split of Normal and anomalous traffic labels, respectively. The reason for the high number of attack samples and a low number of normal samples is that attack samples were generated randomly. In contrast, regular traffic is repetitive actions performed by the nodes.

As this thesis only focuses on detecting anomalies, the different types of attacks will not be classified; instead, all attacks will count as an anomaly. The labels which can be displayed in table 10 are 1) NORMAL, 2) RESPONSE_ATTACK 3) WRITE_ATTACK 4) READ_ATTACK 4) MITM_UNALTERED 5) RECOGNITION_ATTACK 6) FORCE_ERROR_ATTACK 7) REPLAY_ATTACK

Label 1 represents regular network traffic. Labels 2-5 represent injection attacks. Label 6 represents reconnaissance attacks, and label 7 represents replay attacks.

Table 10 displays the number of attack samples after removing the duplicates. The table shows significantly fewer replay and force-error attacks than the original dataset. Removing duplicates also makes the time attribute substantially different from the original dataset, as the time between actions can be considerable compared to in a general control network. Since this does not match patterns of ICS networks leads to the removal when training the machine learning algorithms.

Table 10: Electra labels before and after removing duplicates

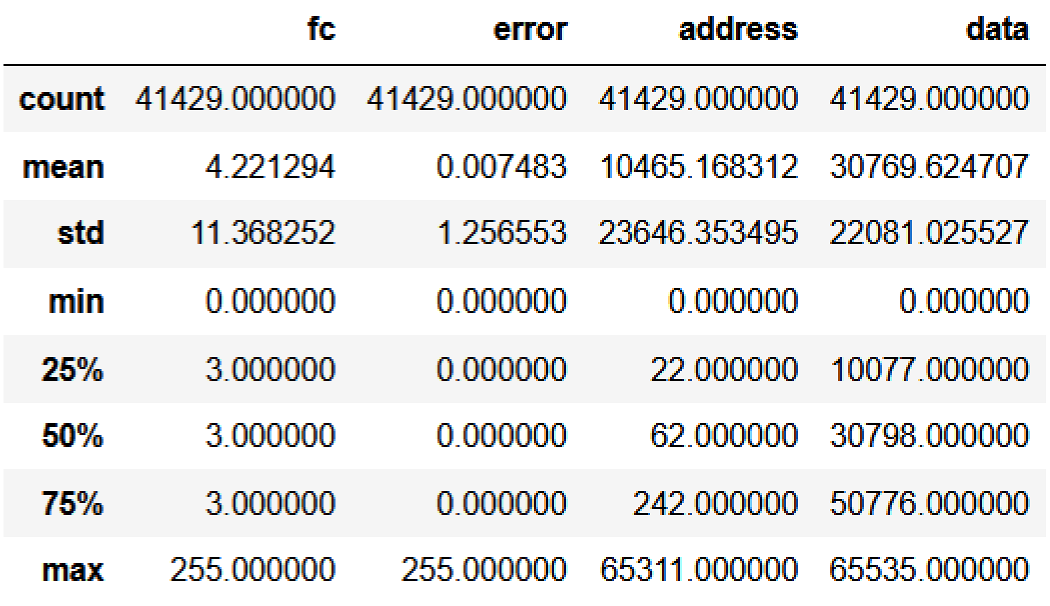
Electra label distribution		Electra no duplicate label distribution	
NORMAL	13894323	NORMAL	24409
MITM_UNALTERED	1550617	RESPONSE_ATTACK	6762
READ_ATTACK	785961	WRITE_ATTACK	6710
RECOGNITION_ATTACK	30580	READ_ATTACK	2671
RESPONSE_ATTACK	16353	MITM_UNALTERED	613
WRITE_ATTACK	9417	RECOGNITION_ATTACK	252
FORCE_ERROR_ATTACK	1129	FORCE_ERROR_ATTACK	10
REPLAY_ATTACK	897	REPLAY_ATTACK	2
Name: label, dtype: int64		Name: label, dtype: int64	

7.4.10 Electra Modbus no duplicates, numerical description

Figure 12 displays a detailed description of the numerical attributes in the no duplicate Electra Modbus. In the figure, the mean is the mean for the feature. Std is the standard deviation. “Min is the lowest value. The 25%, 50%, and 75% rows show the corresponding percentiles: a percentile indicates the value below which a given percentage of observations in a group of observations falls. These are often called the25thpercentile (or 1st quartile), the median, and the 75th percentile (or 3rd quartile)” (Gèron, 2019).

The function code (Fc) and error attribute have a min of 0 and a max of 225. These numbers are not surprising. As explained in the protocol chapter, these are 8-bit binary values. These values have a low mean and standard deviation. From the 1st quartile, median, and 3rd quartile, we can see that Fc and error data have a majority of 3 and 0, respectively.

The address and data field have a high standard deviation. The mean and median are similar, indicating more evenly distributed data. Figure 13 displays the distribution of the data.



	fc	error	address	data
count	41429.000000	41429.000000	41429.000000	41429.000000
mean	4.221294	0.007483	10465.168312	30769.624707
std	11.368252	1.256553	23646.353495	22081.025527
min	0.000000	0.000000	0.000000	0.000000
25%	3.000000	0.000000	22.000000	10077.000000
50%	3.000000	0.000000	62.000000	30798.000000
75%	3.000000	0.000000	242.000000	50776.000000
max	255.000000	255.000000	65311.000000	65535.000000

Figure 12: Description of numerical values in Electra Modbus

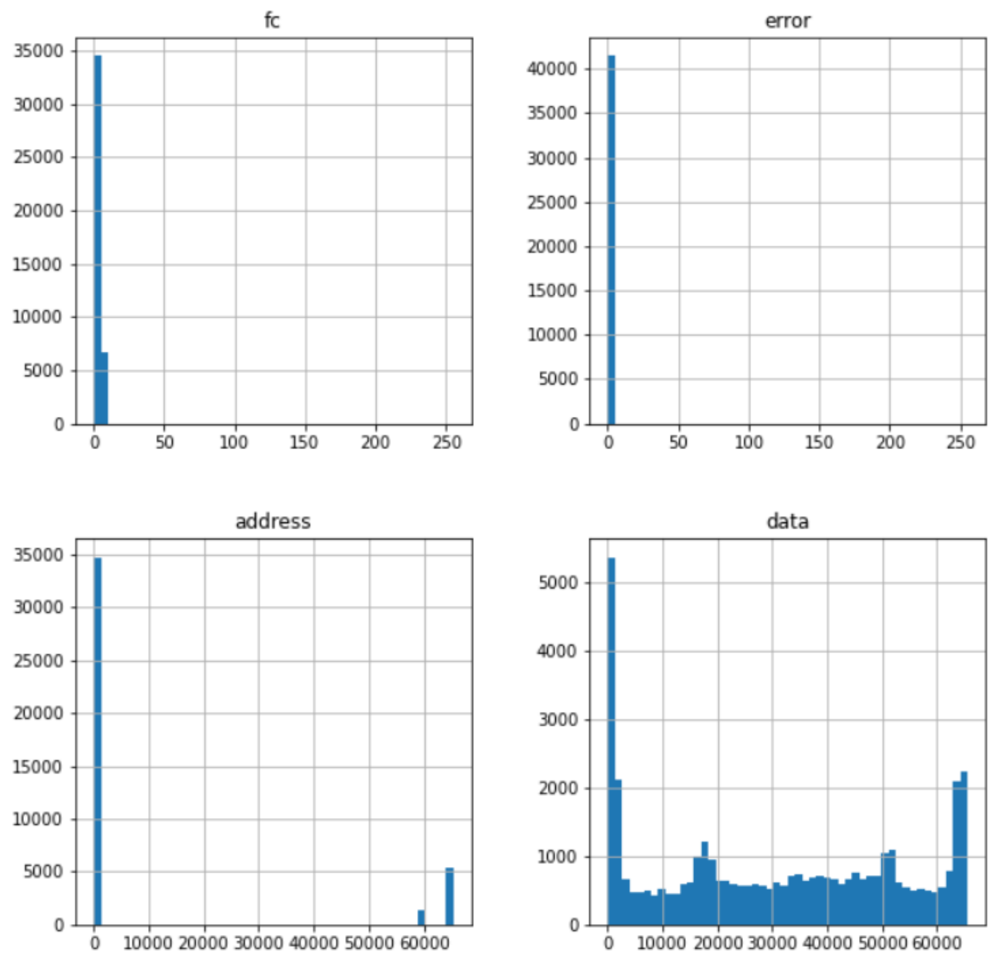
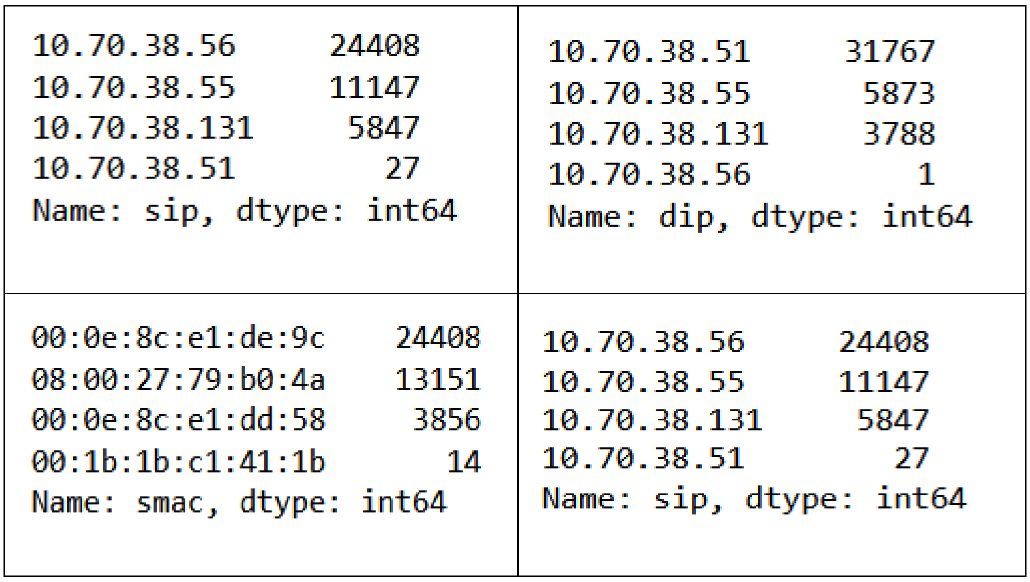


Figure 13: Distribution of Electra Modbus numerical data

7.4.11 Electra object description

Table 11 displays the categorical values in the Electra dataset without duplicate values. When we removed the duplicate data, there might be a biased toward the less used IP and mac addresses. An unsupervised machine learning algorithm might classify these as outliers based on the low number of instances.

Table 11: Categorical objects in Electra Modbus.



```
10.70.38.56      24408
10.70.38.55      11147
10.70.38.131      5847
10.70.38.51         27
Name: sip, dtype: int64
```

```
10.70.38.51      31767
10.70.38.55       5873
10.70.38.131      3788
10.70.38.56          1
Name: dip, dtype: int64
```

```
00:0e:8c:e1:de:9c    24408
08:00:27:79:b0:4a    13151
00:0e:8c:e1:dd:58     3856
00:1b:1b:c1:41:1b       14
Name: smac, dtype: int64
```

```
10.70.38.56      24408
10.70.38.55      11147
10.70.38.131      5847
10.70.38.51         27
Name: sip, dtype: int64
```

7.4.12 Electra test set

Compared to the NSL-KDD dataset, Electra Modbus does not include training data. We split the remaining no duplicate dataset into training and testing. Table 12 shows the even distribution of the attacks. However, compared to the NSL-KDD dataset, this dataset does not include any unknown attacks in the test set. Machine learning algorithms might have higher metrics as they don’t need to generalize new attack types. Table 12 displays the attack distribution in training and testing after removing duplicates.

Table 12: Distribution of data in Electra Modbus with no duplicate records

	Total	Normal	Reconnaissance	False data injection	Replay
Train	33143	19527 (59%)	2339 (7%)	11275 (34%)	2
Test	8286	4882 (59%)	584 (7%)	2819 (34%)	0

7.5 Preparing the data

7.5.1 One hot encoding

Preprocessing must be done on both datasets as computers cannot process categorical data. One hot encoding is a form of vector where all the values except one are 0. 1 represents the category the value belongs means. [0, 0, 1] or [0, 1, 0] are examples of one hot vector where there are three categories of data.

7.5.2 Numeric feature scaling

Feature scaling is a crucial step when preprocessing the data. Scaling is needed for the numerical values to have the same weight on the machine learning outcome. Scaling can differentiate between a weak machine learning model and a better one (Roy, 2020). There are typically two techniques used when applying feature scaling. There are a few different scalers to choose from they are 1) Min-Max scaler, 2) Standard scaler, 3) Max Abs scaler, 4) Robust scaler, 5) Quantile Transformer scaler, 6) Power transformer scaler, 7) Unit vector scaler.

Based on the description by (Roy, 2020) of the weaknesses and strengths of the different scalers. We applied the Robust scaler to the dataset numerical values. Considering the data should contain outliers, the mean and standard deviation difference is high.

7.5.3 Preprocessing the datasets

To keep the experiment consistent. We applied the same preprocessing steps to bough datasets. Firstly, numerical features were scaled using the robust scaler. Secondly, we used a one-hot-encoder to preprocess categorical data. In addition, we removed the time attribute from the Electra Modbus dataset. After preprocessing, the Electra Modbus dataset went from 9 inputs to 22. NSL-KDD when from 41 to 122.

8 Results

Results display the precision metrics of the machine and deep learning models. The different deep learning model's hyper-parameter is kept similar to explore the research questions of the thesis. Deep learning algorithms try multiple hyper-parameters—tables 13 and 14 display the evaluation metrics for LSTM on both datasets. Tables 15 and 16 display evaluation metrics for AE on both datasets. Table 17 displays the best-performing models of both LSTM and AE.

The hyper-parameter for machine learning models is not tuned. As (Gòmez, et al., 2019) have tried numerous hyper-parameters for the machine learning models. The best hyper-parameter was applied to NSL-KDD to compare. Table 18 displays the results from (Gòmez, et al., 2019). Table 19 displays the same machine learning models on NSL-KDD.

Tables [13-19] show a significant difference between the performance evaluations for control and traditional network traffic. These results indicate that machine learning performs better on control networks. However, there are other things to evaluate than the learners' performance. We discuss these shortcomings of the experiment in the discussion chapter.

Table 13: Deep learning algorithms with the best result for Electra Modbus and NSL-KDD

Dataset	Model	Hyper-parameter	Precision	Recall	F1-Score	Accuracy
Electra Modbus	LSTM	Number of layers: 3 Neurons pr layer: [128]	100%	99.68%	99.84%	99.87%
Electra Modbus	Autoencoder	Encoding dim: 11 (50% of input dim) Encoding layers: 1	100%	85.63%	92.26%	94.10%
NSL-KDD	LSTM	Number of layers: 1 Neurons pr layer: [64]	99.76%	43.94%	61.00%	45.06%
NSL-KDD	Autoencoder	Encoding dim: 11 (50% of input dim) Encoding layers: 1	49.27%	94.42%	64.75%	55.71%

In table 14, when optimizing the LSTM on Electra Modbus data, the model recall is generally very high, almost always 100%, meaning that the model detects all the attack data. Precision, however, has a broader fluctuation from 70%-100%. Much of the expected data is classified as attacks for low-precision models. The results from the LSTM models with one hidden layer are generally the same. The best results of the LSTM model are the ones with three layers and the 128 and 64 neurons per layer. Too many layers, however, can lead to overfitting, which might have been the case here considering the low amount of features in the dataset.

Table 14: LSTM with different hyper-parameter applied to Electra Modbus

Hyper-parameter	Precision	Recall	F1-score	Accuracy
Number of layers: 1 Neurons pr layer: 128	70.27%	100%	82.55%	87.78%
Number of layers: 1 Neurons pr layer: 64	70.29%	100%	82.56%	87.79%
Number of layers: 1 Neurons pr layer: 32	70.19%	100%	82.51%	87.74%
Number of layers: 2 Neurons pr layer: 128	78.20%	100%	87.76%	91.04%
Number of layers: 2 Neurons pr layer: 64	78.93%	100%	88.23%	91.34%
Number of layers: 2 Neurons pr layer: 32	74.62%	100%	85.46%	89.57%
Number of layers: 3 Neurons pr layer: 128	100%	99.68%	99.84%	99.87%
Number of layers: 3 Neurons pr layer: 64	93.89%	99.66%	96.69%	97.35%
Number of layers: 3 Neurons pr layer: 32	66.45%	100%	79.84%	86.22%

In table 15, when optimizing the LSTM on NSL-KDD, we can see that the results are significantly lower than that of the Electra optimization. Two models have outlier results, namely the model with 32 neurons per layer and one and two layers. As the input dimension of NSL-KDD is so high, the low number of neurons might have caused the model to underfit. We disregard these models. From the rest of the models, we see that there is a low difference in performance. The low difference indicates that we should have selected a wider variety of model parameters. The accuracy of these models is relatively low too. Since 57% of the data is an anomaly, it would give better accuracy if classifying all attacks as intrusions. The low accuracy score can directly indicate overfitting done in training, showing that the models do not detect the new attacks in the test set.

Table 15: LSTM with different hyper-parameter applied to NSL-KDD

Hyper-parameter	Precision	Recall	F1-score	Accuracy
Number of layers: 1 Neurons pr layer: 128	99.83%	44.33%	60.44%	43.71%
Number of layers: 1 Neurons pr layer: 64	99.76%	43.94%	61.00%	45.06%
Number of layers: 1 Neurons pr layer: 32	0.08%	2.15%	0.15%	55.35%
Number of layers: 2 Neurons pr layer: 128	99.47%	43.93%	60.99%	45.05%
Number of layers: 2 Neurons pr layer: 64	99.82%	43.94%	61.02%	45.08%
Number of layers: 2 Neurons pr layer: 32	0.04%	2.7%	0.01%	56.3%
Number of layers: 3 Neurons pr layer: 128	99.73%	43.93%	60.99%	45.05%
Number of layers: 3 Neurons pr layer: 64	99.76%	43.93%	61.00%	45.06%
Number of layers: 3 Neurons pr layer: 32	99.71%	43.92%	60.98%	45.04%

Tables 16 and 17 show the optimization of the AE. Even though the f1-score of all the modes was not as high as the best LSTM, the AE scored better than most LSTMs for the Electra dataset. The AE recall was also better across the board, meaning the learner did not send as many false alarms. On the NSL-KDD dataset, the AE did outperform the LSTM. The recall of the AE was low on NSL-KDD, indicating the AE sent a low number of false alarms. However, looking at the accuracy, even though the AE did perform better, it performed worse than classifying all the data as malicious. The low accuracy indicates overfitting the training set and cannot generalize on new attack instances.

Table 16: Autoencoder with different hyper-parameter applied to Electra Modbus

Hyper-parameter	Precision	Recall	F1-score	Accuracy
Encoding dim: 16 (75% of input dim) Encoding layers: 1	100%	85.31%	92.07%	93.96%
Encoding dim: 11 (50% of input dim) Encoding layers: 1	100%	85.63%	92.26%	94.10%
Encoding dim: 5 (25% of input dim) Encoding layers: 1	100%	85.23%	92.07	93.85%

Table 17: Autoencoder with different hyper-parameter applied to NSL-KDD

Hyper-parameter	Precision	Recall	F1-score	Accuracy
Encoding dim: 16 (75% input dim) Encoding layers: 1	48.87%	94.48%	64.42%	55.05%
Encoding dim: 11 (50% of input dim) Encoding layers: 1	49.19%	94.48%	64.70%	55.59%
Encoding dim: 5 (25% of input dim) Encoding layers: 1	49.27%	94.42%	64.75%	55.71%

Table 19 displays the machine learning result for NSL-KDD. Even though they performed worse than the ones in table 18 by (Perales Gòmez, et al., 2019), they did perform better than the DL models (except for isolation forest) in terms of f1-score. The relatively low precision for these ML models indicates that many attacks would have gone undetected in NSL-KDD. Worth considering is the fact that the ML models on NSL-KDD were not hyper-parameter optimized. With other hyper-parameter, these models might have performed significantly better. We should also have measured the accuracy of these models to detect overfitting.

Table 18: Machine learning algorithms applied to Electra Modbus (Gòmez, et al., 2019)

Dataset	Model	Hyper-parameters	Precision	Recall	F1-Score
Electra Modbus	Random Forest	Estimator: 200	98.77%	98.71%	98.74%
	SVM	C: 10 Gamma: 1	97.56%	100%	98.76%
	Artificial Neural Network	Nr of layers: 1 Neurons per layer: [128]	96.92%	100%	98.43%
	OCSVM	Nu: 0.1 Gamma: 0.1	98.62%	98.56%	98.59%
	Isolation Forest	Estimators: 100 Contamination: 0.1	87.39%	100%	93.27%

Table 19: Machine learning algorithms applied to NSL-KDD

Dataset	Model	Hyper-parameters	Precision	Recall	F1-Score
NSL-KDD	Random Forest	Estimator: 200	63.38%	90.09%	74.41%
	SVM	C: 10 Gamma: 1	61.47%	97.25%	75.33%
	Artificial Neural Network	Nr of layers: 1 Neurons per layer: [128]	62.08%	90.30%	73.57%
	OCSVM	Nu: 0.1 Gamma: 0.1	78.00%	89.47%	83.34%
	Isolation Forest	Estimators: 100 Contamination: 0.1	31.63%	83.93%	45.94%

9 Discussions

Even though it might seem that research question one, "*Is it more suitable to apply machine learning in industrial networks compared to in traditional networks*?" has a definite answer because of the significate difference in performance. There are other aspects of machine learning that can have an impact on the measured performances. In this chapter, we will discuss these features affecting performance.

9.1 Limitations

The landscape of Industrial Control Systems is significant, and there are multiple types of Industrial Control systems. We selected the Supervisory Control and Automation type of Industrial Control System. Intrusion detection systems are also a broad landscape. The thesis mainly focuses on the ones used in the network and used machine learning.

A limitation of the results research questions for the thesis is the availability of datasets used for machine learning in Network Intrusion Detection Systems. There are many benchmark datasets available for traditional networks, however. Because of this, many machine learning algorithms in the datasets there are machine learning algorithms applied to these individual datasets. As this is not the case for industrial networks, there are fewer machine learning algorithms to compare. The thesis focuses mainly on the supervised- and unsupervised learning models.

9.2 Dataset fundamentals

Traditional networks have a variety of benchmark datasets captured from real scenarios. ICS does not have this luxury. Machine learning algorithms are only as valuable as the data used—garbage in, garbage out. (Wang & Foo, 2018) proposes a structured approach to evaluating ICS datasets with fundamental, special, and reality requirements.

The fundamental requirements that affect the learning algorithms' performance include training, testing correlation, and capture from all nodes.

Both datasets have a high correlation between training and testing for training testing. Tables 5 and 12 show this. However, the NSL-KDD dataset has unique attacks in the testing set, which requires the learning algorithm to generalize well for high-performance measurements. The unique attack is the training set is part of what leads to a difference in performance between the learning algorithms.

Special requirements mentioned by (Wang & Foo, 2018) are about regular and abnormal traffic. These can directly affect the learner's performance—a special requirement is the number of nodes captured in the network traffic. Electra Modbus dataset only uses a small portion of the nodes available. The small number of communicating nodes makes the Electra Modbus dataset less realistic than general control network traffic. In a more natural environment, as seen in the ICS chapter, many more devices are often required to communicate to perform a task.

Electra Modbus only uses one communication protocol for the expected traffic. As the IDS chapter explains, NIDS should cover a broader range of hosts. Therefore, it is generally better to have multiple protocols in the datasets. Including one protocol also affects the attack surface as some attacks use other protocol vulnerabilities. Considering this combining the Modbus and S7Comm would have given more realistic results.

The anomaly samples for Electra datasets with no duplicates, displayed in table 12, are unevenly distributed. In the table, we see a far higher percentage of repaying attacks which can contribute to the high performance of the learners.

9.3 Feature selection

This experiment compares machine learning algorithms against each other. As shown in table 7, NSL-KDD, compared to Electra Modbus, includes a variety of protocols, services, and TCP flags. Because of this, during preprocessing, the NSL-KDD dataset gets a significantly more considerable amount of input features. As shown in the work by (K. A. Taher, 2019) and (L. Hakim, 2019), the features selected significantly impact the learner. Considering this, we should only have included directly correlated features between the datasets. Table 2 of the OSI model would feature 1-4 as these layers are similar in traditional and control traffic. These features are, however, not included in both datasets. In the future, when comparing traditional- and control networks. Datasets should consist of full packet capture, selecting only these features. Another approach to the mismatch of the attributes in conventional- and control datasets would be to include log features such as those described by (BDO AS, 2014). The traditional benchmark datasets often include data collected from logs. Creators of control network datasets should contain log files for comparing traditional- and control network traffic. For intrusion detection on ICS datasets (Wang & Foo, 2018) also request this.

9.4 Biases

To compare the tradition- and control network dataset, this thesis conducted a literature study of ICS that might have biased the learners' results. Electra's numerical features were in focus when selecting the numerical scaler for the dataset. However, in the literature, many other normalizations and scaling techniques are to NSL-KDD. Focusing heavily on the ICS datasets has led to biases in the results. As shown in the work of (Umar & Zhanfang, 2020), the selection of normalization or scaling directly affects the impact of the learner. The numeric scaler was kept consistent for the sake of comparing the learner. However, future work comparing datasets should thoroughly analyze all numerical features. If the features from both datasets were the same, the impact of this might not have been that large.

Another bias toward the control network dataset is the hyperparameter of the learns. Traditional machine learner's hyper-parameters were selected based on the work of (Perales Gòmez, et al., 2019), which optimized the hyper-parameters for the traditional learner based on the control network dataset. For example, the work of (Meira, 2019) and (Ingre & Yadav, 2015) shows a higher performance evaluation of Isolation forests and ANN on NSL-KDD, respectively.

10 Conclusion

This thesis went through a literature study of ICS and ML- and DL methods used in IDS to determine if these are better suited in control networks as these environments are more closed to human input. We applied various ML- and DL methods to NSL-KDD and Electra Modbus. We selected datasets based on the coverage of attacks trying to keep these comparable even though attack vectors for the different networks vary.

To answer the first question, “Is it more suitable to apply machine learning in industrial networks compared to traditional networks?” this thesis applied LSTM and AE to both traditional and control networks as a NIDS detection engine. In addition, we selected a range of machine learning algorithms to use on the NSL-KDD dataset. The resulting experiment shows significantly higher results on the control network traffic than on traditional traffic. The f1-score of the LSTM was 99.84% and 61.00% on the control and traditional network, respectively. The f1-score of the autoencoder was 92.26% and 64.75% on the control- and traditional-network, respectively. All the machine algorithms applied on Electra Modbus outperformed the one used on NSL-KDD. However, an important factor when evaluating machine learning is the datasets. For traditional network traffic, there is a wide array of benchmark datasets. However, for control-network traffic, there is no such benchmark. To evaluate if machine learning is more suitable in industrial networks without a benchmark dataset therefore inconclusive.

The second research question derived from this experiment, “i*s deep learning more powerful in detection intrusions than traditional machine learning methods?”* is also hard to evaluate because of the lack of benchmark datasets. From the experiment conducted, one of the deep learning models, the LSTM, performed better with an f1-score of 99.84%, which is better than the best machine learning model (Gòmez, et al., 2019) with an f1-score of 98.74%. The Autoencoder, on the other hand, performed significantly worse with an f1-score of 92.26%. The ML methods on NSL-KDD all performed better than the DL methods, except isolation forest.

10.1 Future work

Future work requires a benchmark dataset for SCADA systems. This dataset should contain network traffic in addition to log files should be included to extract features that are compatible with traditional network features. The SCADA system network data should also be captured from more devices and have a variety of protocols for the dataset to be considered a benchmark.

A structured approach for comparing the datasets is also required to remove biases from the comparison. This approach should weigh both datasets' feature selection, isolating the same features, to get comparable results. The learns hyper-parameters should also be randomly selected to avoid biases.

11 Bibliografi

Acromag. (2005). *INTRODUCTION TO MODBUS TCP/IP.* ACROMAG INCORPORATED.

Adepu, S., Kandasamy, N. K., & Mathur, A. (2019). Epic: An electric power testbed for research and training in cyber physical systems security. *Computers Security* (ss. 37-52). Cham, Switzerland: Springer.

Ahmed, C. M., Parkash, J., & Zhou, J. (2020). *Revisiting Anomaly Detection in ICS: Aimed at Segregation of Attacks and Faults.* Singapore: ResearchGate.

Alimi, O. A., Ouahada, K., Abu-Mahfouz, A. M., Rimer, S., & Alimi, K. O. (2021). *A Review of Research Works on Supervised Learning Algorithms for SCADA Intrusion Detection and Classification.* sustainability.

Anderson, J. P. (1980). *Computer Security Threat Monitoring and Surveillance.* Fort Washington.

Aygun, R. C., & Yavuz, A. G. (2017). *Network Anomaly Detection with Stochastically Improved Autoencoder Based.* Istanbul, Turkey: IEEE.

Babu, B., Ijyas, T., P, M., & Varghese, J. (2017). *Security Issues in SCADA based Industrial Control Systems.* abha: IEEE.

Baezner, M., & Robin, P. (2017). *Stuxnet.* ETH Zürich: Center for Security Studies (CSS).

BDO AS. (2014). *Metodikk for informasjonsinnhenting etter IKT-sikkerhetshendelser i.* Norges vassdrags- og energidirektorat.

Beaver, J. M., Borges-Hink, R. C., & Buckner, M. A. (2013). An evaluation of machine learning methods to detect malicious SCADA communications. (ss. 54–59).

Bosijancic Rakas, S. V., Stojanovic, M. D., & Markovic-Petrovic, J. D. (2020). *A Review of Research Work on Network-Based SCADA Intrusion Detection Systems.* IEEE.

Campbell, M., Hoane, J. A., & Feng-hsiung, H. (2002). *Deep Blue.* New York: Elsevier B.V.

Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly Detection: A Survey. *ACM Computing Surveys* (ss. 1-12). Minneapolis: DBLP.

CICIDS. (2011). Hentet fra https://www.stratosphereips.org/datasets-ctu13

Collantes, M. H., & Padilla, A. L. (2015). *Protocols and network security in ICS infrastructures.* Incibe.

Darpa98. (1998, Feb). Hentet fra https://www.ll.mit.edu/r-d/datasets/1998-darpa-intrusion-detection-evaluation-dataset

Dertat, A. (2017, Oct 3). *towardsdatascience.* towardsdatascience.com: https://towardsdatascience.com/applied-deep-learning-part-3-autoencoders-1c083af4d798

Dev, R. R., & Abualkibash, M. (2019). *INTRUSION DETECTION SYSTEM CLASSIFICATION USING DIFFERENT MACHINE LEARNING ALGORITHMS ON KDD-99 AND NSL-KDD DATASETS.* Ypsilanti, Michigan, USA: International Journal of Computer Science & Information Technology (IJCSIT).

Dhanabal, L., & Shantharajah, S. P. (2015). *A Study on NSL-KDD Dataset for Intrusion Detection System Based on Classification Algorithms.* Coimbatore, India: International Journal of Advanced Research in Computer and Communication Engineering.

Ever, Y. K., Sekeroglu, B., & Dimililer , K. (2019). *Classification Analysis of Intrusion Detection on NSL-KDD Using Machine Learning Algorithms.* Springer .

Fan, J., Cong, M., & Zhong, Y. (2021). *A Selective Overview of Deep Learning.* Institute of Mathematical Statistics.

Feng, C., Li, T., & Chana, D. (2017). *Multi-level Anomaly Detection in industrial Control Systems via package Signatures and LSTM Networks.* Denver: IEEE.

(2014). *Framework for Improving.* National Institute of Standards and Technology.

Gao, J., Gan, L., Buschendorf, F., Zhang, L., Liu, H., Li, P., . . . Lu, T. (2019). *LSTM for SCADA Intrusion Detection.* Victoria, Canada: IEEE.

Gèron, A. (2019). *Hands-on Machine Learning with Scikit-learn, Keras & TensorFlow.* Canada: O'Reilly Media, Inc.

Gòmez, À. L., Celdràn, A. H., Sarmiento, C. C., Maimò, L. F., Clemente, F. J., Del Canto Masa, C. J., & Nistal, M. N. (2019). *On the Generation of Anomaly Detection Datasets in Industrial Control Systems.* Murica, Waterford: IEEE.

Gonzalez, D., Alhenaki, F., & Mirakhorli, M. (2019). *Architectural Security Weaknesses in Industrial.* Rochester, NY USA: IEEE.

Holländer, B. (2020, October 1). *medium.com*. Hentet fra medium.com: https://medium.com/p/86135f7c0d35

Huntchins, E. M., Cloppert, M. J., & Amin, R. M. (2011). *Intelligence-Driven Computer Network Defense Informed by Analysis of Adversary Campaigns and Intrusion Kill Chains.* ResearchGate.

Hutchins, E. M., Cloppert, M. J., & Amin, R. M. (2011). *Intelligence-Driven Computer Network Defense Informed by Analysis of Adversary Campaigns and Intrusion Kill Chains.* ResearchGate.

Hvoland, K. (2017). *Logging og logganalyse i energiforsyningen.* Oslo: Norges vassdrags- og energidirektorat.

Ingre, B., & Yadav, A. (2015). *Performance analysis of NSL-KDD dataset using ANN.* Guntur, India: IEE.

ITrust. (2015). Hentet fra https://itrust.sutd.edu.sg/itrust-labs_datasets/dataset_info/

ITrust. (2019, Dec). Hentet fra https://itrust.sutd.edu.sg/itrust-labs_datasets/dataset_info/

Johnson, A. L. (2014, Jun 8). *community.broadcom.com*. https://community.broadcom.com/symantecenterprise/communities/community-home/librarydocuments/viewdocument?DocumentKey=7382dce7-0260-4782-84cc-890971ed3f17&CommunityKey=1ecf5f55-9545-44d6-b0f4-4e4a7f5f5e68&tab=librarydocuments

Joshi, N. (2017). Machine learning for anomaly detection.

Joyothsna, V., Rama Prasad, V. V., & Munivara Prasad, K. (2011). *A Review of Anomaly based intrusion Detection Systems.* Tirupati: International Journal of Computer Applications.

K. A. Taher, B. M. (2019). K. A. Taher, B. Mohammed Yasin Jisan and M. M. Rahman. *International Conference on Robotics,Electrical and Signal Processing Techniques (ICREST)* (ss. 643-646). Dhaka, Bangladesh: IEEE.

Kaspersky Lab. (2021). *Threat landscape for auomation systems.* Kaspersky ICS Cert.

KDD99. (u.d.). Hentet fra http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

Kim, S., Jo, W., & Shon, T. (2019). *APAD: Autoencoder-based Payload Anomaly Detection for industrial IoE.* Suwon-si: ScoemceDorect.

Knapp, E. (2011). *Industrial Network Security.* Sciencedirect.

Knapp, E. D., & Langill, J. T. (2014). *Industrial Network Security, 2nd Edition.* Syngress.

L. Hakim, R. F. (2019). Influence Analysis of Feature Selection to Network Intrusion Detection System Performance Using NSL-KDD Dataset. *International Conference on Computer Science, Information Technology, and Electrical Engineering (ICOMITEE)* (ss. 217-220). Jember, Indonesia: IEEE.

M System co ltd. (u.d.). *Modbus Protocol Reference Guide.* Japan: M system co ltd.

M, Z., & Movahedi, M. (2015). *Machine learning techniques for intrusion detection.* CoRR.

Malviya, N. (2019). ICS Protocols.

Mantere, M., Sailio, M., & Noponen, S. (2015). *A Module for Anomaly Detection in ICS Networks.* Oulu: ACM.

Mathur, A. P., & Tippenhauer, N. O. (2016). Swat: A water treatment testbed for. *Proc. Int. Workshop Cyber-Phys*, (ss. 31–36).

McClanahan, R. H. (2002). *The benefits of networked SCADA systems utilizing IP-enabled networks.* Colorado Springs: IEEE.

Meira, J. A. (2019). Performance evaluation of unsupervised techniques in cyber-attack anomaly detection. *Ambient Intell Human Comput* (ss. 4477–4489). Springer link.

Meng, Y.-X. (2011). The practice on using machine learning for network anomaly intrusion detection. *International Conference on Machine Learning and Cybernetics.* Guilin, China: IEEE.

MODICON, Inc., Industrial Automation Systems. (1996). *Modicon Modbus Protocol Reference Guide.* North Andover, Massachusetts: MODICON, Inc., Industrial Automation Systems.

Monowar, B. H., Bhattacharyya, D. K., & Kalita, J. K. (2014). *Network Anomaly Detection: Methods, Systems and tools.* IEEE.

Morris, T., Srivastava, A., Reaves, B., & Gao, W. K. (2011). A control system testbed to validate critical infrastructure protection concepts. *Int. J. Crit. Infrastruct. Protection*, (ss. 88–103).

Negandhi, P., Trivedi, Y., & Mangrulkar, R. (2019). Intrusion Detection System Using Random Forest on the NSL-KDD Dataset. I *Emerging Research in Computing, Information, Communication and Applications* (ss. 519–531). Singapore: Springer.

NGIDS-DS. (2016). Hentet fra https://research.unsw.edu.au/people/professor-jiankun-hu

Ning, P., & Jajodia, S. (2003). *Intrusion Detection Techniques.* The Internet Encyclopedia.

NSL-KDD. (2015). *unb.ca*. Hentet fra https://www.unb.ca/cic/datasets/nsl.html: https://www.unb.ca/cic/datasets/nsl.html

Pan, S., Morris, T., & Adhikari, U. (2015). Developing a Hybrid Intrusion Detection System Using Data Mining for Power Systems. *IEEE Transactions on Smart Grid*, (ss. 3104-3113).

Parkhi, O., Vedaldi, A., & Zisserman, A. (2015). *Deep face recognition.* Oxford: British Machine Vision Association.

Perales Gòmez, À. L., Maimò, L. F., Celdràn, A. H., Clemente, F. J., Sarmiento, C. C., Del Canto Masa, C. J., & Nistal, R. M. (2019). *On the Generation of Anomaly Detection Dataset in Industial Control Systems.* Murcia: IEEE.

Protić, D. D. (2018). *REVIEW OF KDD CUP '99, NSL-KDD AND KYOTO 2006+ DATASETS.* Belgrade, Republic of Serbia.

Qassim, Q. S., Jamil, N., Patel, A., & Ja'affar, N. (2019). *A review of security assessment methodologies in industrial control systems.* Selangor: Emerald insight.

Roopa Devi, E. M., & Suganthe, R. C. (2018). *Enhanced transductive support vector machine classification with grey wolf optimizer cuckoo search optimization for intrusion detection system.*

Roy, B. (2020, April 6.). *towardsdatascience*. towardsdatascience: https://towardsdatascience.com/all-about-feature-scaling-bcc0ad75cb35

Saporito, G. (2019, Sep 17.). *towardsdatascience*. Hentet fra https://towardsdatascience.com/a-deeper-dive-into-the-nsl-kdd-data-set-15c753364657

Sewak, M., Sahay, S. K., & Rathore, H. (2022). Deep Reinforcement Learning for Cybersecurity Threat Detection and Protection: A Review. *Communications in Computer and Information Science* (ss. 51–72). Spring Link.

Shrestga, A., & Mahmood, A. (2019). *Review of Deep Learning Algorithms.* Bridgeport USA: IEEE.

Singh, S., & Sanjay, S. (2015). *Cyber Attack Detection System based on Improved Support Vector Machine.* International Journal of Security and Its Applications.

Slowik, J. (2019). *Evolution of ICS Attacks and the Prospects.*

Smith, J. (2016). GitHub: https://github.com/automayt/ICS-pcap

Sokolov, A. N., Pyatnitsky, I. A., & Alabugin, S. K. (2019). *Applying Methods of Machine Learning in the Task of Intrusion Detection Based on the Analysis of Industrial Process State and ICS Networking.* Chelyabinsk: Imprint.

Sommer, R., & Paxson, V. (2010). *Outside the Closed World: On Using Machine Learning For Network Intrusion Detection.* Berkeley: IEEE computer society.

Tasi, C.-F., Hsu, Y.-F., Lin, C.-Y., & Lin, W.-Y. (2009). *Intrusion Detection by machine learning: A review.* Taiwan: Elsevier Ltd.

Team, K. L. (2014). *Energetic Bear — Crouching Yet.* Kaspersky Lab Global Research and Analysis Team. Hentet fra https://media.kasperskycontenthub.com/wp-content/uploads/sites/43/2018/03/08080817/EB-YetiJuly2014-Public.pdf

Umar, M. A., & Zhanfang, C. (2020). *Effects of Feature Selection and Normalization on Network.* 7186 Weixing Road, Jilin, China.

UNSW-NB15. (2021). https://www.unsw.adfa.edu.au/unsw-canberra-cyber/cybersecurity/ADFA-NB15-Datasets/

Varsamopoulos, S., & Bertels, K. (2018). *Designing neural network based decoders for surface codes.* ResearchGate.

Wang, X., & Foo, E. (2018). *Assessing Industrial Control System Attack Datasets for Intrusion Detection.* Brisbane, Australia: IEEE.

Water Distribution (Wadi). (2015). Hentet fra https://itrust.sutd.edu.sg/testbeds/water-distribution-wadi/

Wun-Hwa, C., Sheng-Hsun, H., & Hwang-Pin, S. (2005). *Application of SVM and ANN for intrusion detection.* Science direct.

Xiaoyong, Y., Chuanhuang, L., & Xiaolin, L. (2017). *DeepDefence: Identifying DDoS attacks via deep learning.* Florida, USA: IEEE.

Yadav, G., & Paul, K. (2020). *Architecture and Security of SCADA Systems: A Review.* Delhi: ResearchGate.

YIN, C., YUEFEI, Z., JINLONG, F., & XINZHENG, H. (2017). *A Deep Learning Approach for Intrusion detection using recurrent reural networks.* Zhengzhou, China: IEEE.

ZAREI, K. (2020, Feb 26). *norcalcontrols.net*. Hentet fra blog.norcalcontrols.net: https://blog.norcalcontrols.net/scada-networking-protocols-and-basics

Zeeshan, A., Adnan, S. K., Cheah, W. S., Johari, A., & Farhan, A. (2021). *Network intrusion detection system: A systematic study of machine learning and deep learning approaches.* Trans Emerging Tel Tech.

Zhang, R. B., Xia, L. H., & Lu, Y. (2019). *Anomaly Detection of ICS based on EB-OCSVM.* Anhui: IOP Publishing.