Jan Olaf Storeng

# Developing Long Short-Term Memory Mechanism and Dataset for Intrusion Detection in Critical Infrastructure by Simulation

Master's thesis in Information Security
Supervisor: Slobodan Petrovic
June 2024

Master's thesis

**NTNU**
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Dept. of Information Security and Communication Technology

Jan Olaf Storeng

# Developing Long Short-Term Memory Mechanism and Dataset for Intrusion Detection in Critical Infrastructure by Simulation

Master's thesis in Information Security
Supervisor: Slobodan Petrovic
June 2024

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Dept. of Information Security and Communication Technology

# Sammendrag

Beskyttelsen av kritisk infrastruktur er svært viktig for å kunne forsørge privatpersoner og bedrifter med essensielle ressurser. Dette kan være utfordrende grunnet det sensitive og komplekse ICS-miljøet, samt motiverte og kapable aktører med sofistikerte verktøy og ressurser, skreddersydd for systemene. Maskinlæring er bevist å kunne være en effektiv metode for å detektere og hindre forsøk på cyberangrep. Grunnet en mangel på datasett som reflekterer kritisk infrastruktur, vil det kunne være vanskelig å lage modeller relatert til miljøets adferd. Det er derfor gitt forslag til en metodikk som omhandler simulering av miljøet for å kunne blir brukt til utviklingen av en maskinlæringsmodell, for klassifisering av adferd. Metoden tar utgangspunkt i å etterlikne adferden av en elektrisk transformatorstasjon ved å definere scenarioer som reflekterer normal og unormal operasjon. En Long Short-Term Memory modell er trent på datasettet og evaluert etter hvilke scenarioer den klarer å identifisere. Basert på prestering og hvilke egenskaper modellen viser, var modellen forbedret gjennom iterasjoner for til slutt å kunne prestere etter behovet. Ved bruk av et simulert datasett var det observert at den resulterende modellen ble transparent gjennom forståelsen av dataen den var lært på. Dette ga innsyn i hvilke mangler modellen hadde og veiledet videreutviklingen. En utfordring ved simulering er å realistisk kunne reflektere adferden av Industrial Control System. Miljøet som ellers inneholder komplekse forbindelser og relasjoner, er ved simulering redusert til den definerte adferden hvilket ikke i like stor grad spiller på styrkene til maskinlæring. Fremtidig arbeid relateres til videreutvikling av metoder for å forbedre Intrusion Detection System (IDS) i kritisk infrastruktur. For simulering å kunne reflektere presis adferd av kritisk infrastruktur trengs spesifiserte verktøy til de relevante omgivelsene.

# Abstract

The protection of Critical Infrastructure (CI) is significantly important to secure and maintain availability of resources essential for private persons and organisations, nationally. This is challenging due to the scale and complexity of its environment and motivated and capable malicious actors in possession of sophisticated tools and resources, capable of disrupting the services. Machine Learning (ML) has proven to be a capable technology in intrusion detection and prevention for mitigating and stopping attempts of cyberattacks. There is however a lack of relevant dataset for CI, which makes the development of efficient ML models harder. It is for this thesis therefore proposed a methodology to simulate a dataset mimicking the behaviour of CI, for use in development of a ML classification model. The methodology is explored by a use case of developing a dataset depicting normal and abnormal behaviour of an electrical substation. The dataset is simulated by defined scenarios which describes the behaviour of features. A Long Short-Term Memory (LSTM) model is trained on the dataset and evaluated in regards to what scenarios it is capable of identifying. Based on the capabilities and traits of the model it is improved to best depict the simulated environment. Simulation was found to give insight into the inner workings of the developed ML model, providing transparency of its capabilities and weaknesses. This enabled efficient iteration of the LSTM model, enhancing its capabilities and necessary traits to be able to detect the more challenging attack scenarios. Simulating the environment of CI for the purpose of training a ML model, was however observed to be insufficient in representing an Industrial Control System (ICS) environment. Data which otherwise depicts a complex, interconnected network of dependable components are by simulation reduced to highly defined behaviour which does not play into the strengths of deep learning. Future work describes further developing methodologies which support the development of Intrusion Detection System (IDS) in CI. For simulation, this requires tools which can accurately depict the complexity of its environment.

# Acknowledgment

I would like to thank my supervisor of this thesis, Slobodan Petrovic for his support and insight throughout the thesis. His guidance has throughout the period lead me on the right path. The assistance which has been provided has enabled me to deepen my knowledge and reinforcing my interest in the field of information security.

I would also like to thank Ahmed Walid Amro for providing me with his assistance and expertise in helping me traverse subjects of unfamiliarity, generously introducing me to concepts which was necessary for completion of the thesis.

# Preface

The thesis is submitted as the final step of the Msc. in Information Security at the Norwegian University of Science and Technology (NTNU). The thesis was conducted in the period of January 2024 and June 2024, and was 30 credit points. The supervisor of the thesis was Professor Slobodan Petrovic at the department of Information Security and Communication Technology.

# Contents

# List of Figures

## List of Tables

# 1 Introduction

This chapter describes the background, reason and motivation for writing this thesis. First, the introduction gives a brief description of what topics are covered throughout the paper. Thereby the motivation for why the thesis was created is justified, before the problem which is attempted solved is described. Afterwards the research questions are presented together with a description of the planned contribution. Later the structure of the thesis is described before relevant previous work is presented in the literature review. The introduction summarizes key points which are presented in the background, where the sources of the stated insight is also cited.

## 1.1 Covered topics

The Industrial Control System (ICS) describes environments which communicate between physical and digital processes. Data transferred communicate the state of actuators and sensors, determining automated processes. ICS's are found implemented in factories, energy providers, power plants, water treatment plants and are essential for production, maintenance, and distribution. Disruptions in the ICS could have serious implications, impacting both physical components and safety of those dependent on the service. This is especially true for Critical Infrastructure (CI), where distribution and availability of the service is crucial. Both the environment of ICS and CI have been covered in relation to what makes them different from normal networks. The components of the ICS and their functions are described to give a background in what types of data they might generate and what requirements of the environment must be taken into consideration for the development of an Intrusion Detection System (IDS). This knowledge has been used to establish sources for features used in developing a dataset.

In an attempt of securing the environment of CI, it is proposed a methodology for creating a Machine Learning (ML) model for use in IDS. The thesis describes the importance and challenge in utilizing a dataset which accurately portrays the target environment. Shortcomings in "state of the art" datasets are motivation for simulating a dataset rather than make use of existing datasets.

Recent developments and innovations of the area of ICS introduced by Industry 4.0 has benefited the operation and management of the environment. However, due to shortcomings in securing protocols and components used in ICS, the implementation of Industry 4.0 introduces more attack-vectors towards the environment.

IDS's are heavily used to detect malicious behaviour within a network, and are important tools for maintaining a secure state. IDS can provide early warnings of breaches and help limit damages. ML has proven to be useful methodology in detecting malicious behaviour as large amounts of data from different sources are utilized in developing an understanding of the context. Different types of ML-techniques and architectures are introduced to give a foundation of knowledge to find what models should be used for the defined scenario. The thesis will focus on Artificial Neural Network (ANN), more specifically of the Long Short-Term Memory (LSTM) type, while also make use of other ML-techniques for comparison and validation.

## 1.2 Motivation

The importance of maintaining the availability of CI makes it a target for both politically and economically motivated actors. Losses in CI could have severe economical and political consequences, also leading to loss of life. Securing assets from hackers is demanding given their level of experience and access to resources. There is need for reliable ways of detecting malicious activity, even more so in the environment of CI where operational systems are being transitioned into the 4th industrial revolution. The transition has opened the environment for more types of attacks towards devices with limited implementation of security [1]–[6]. *Industry 4.0*, although being beneficial to the management and operation, possesses weaknesses which now are exposed over internet [3]. This could be misused to control and mismanage operations. Components of industry 4.0 make managing and maintaining a secure state harder as there is a conflict of interest between the new

generation of components and the criticality of the environment. Management over 5G, WiFi and other IP-based protocols has opened the environment of CI to the rest of the world. This has opened up a larger number of attack vectors hwic impacts critical components in gaining entrance to the environment.
To be able to detect intrusion attempts in a big data environment, where a wide variety of diverse attacks could be performed, data from multiple sources are found to be beneficial in designing a robust threat-defensive infrastructure [7]. Research performed in this paper is provided to help further detection methodology by presenting an additional source of data and its quality in detecting intrusions.
An important aspect of achieving reliable detection is to create and test the IDS towards a dataset that reflects the environment accurately. There is a need for data that accurately characterizes the CI environment to train and develop IDS for both network and hosts [1]. Most IDS's are based on legacy datasets which do not reflect current network behaviour, making them inapplicable to the system. Therefore in this thesis, a simulated dataset is developed to portray the environment of CI in the form of an electrical substation to find out to what degree the environment can be mimicked. In addition to the dataset, ML models are developed and trained on the dataset in an attempt to classify malicious and benign behaviour of the environment.

## 1.3 Problem description

The security of Critical Infrastructure (CI) is essential to maintain the safety of a nation. Challenges given the environment do however make this difficult. Newly implemented devices and technology of the 4th industrial revolution have expanded the attack surface of the CI-sector. This has introduced more attack methodologies that must be measured and prevented. The environment is also a target of sophisticated attackers with resources, knowledge, tools and motivation to perform attacks which are hard to defend against. Machine learning approaches in detecting malicious behaviour has proven to be a beneficial method of recognizing attacks. Very few machine learning methodologies presented does however represent an actual CI environment. Given the severity in identifying weaknesses in CI environment, there are few datasets that provide necessary insight to its behaviour. As the CI environment functions quite differently from other types of networks, models built on irrelevant datasets are therefore found to be inapplicable.

For this thesis a classification model using Long Short-Term Memory (LSTM) architecture built on a simulated dataset is made. The thesis presents a use case simulating the measurements of an electrical substation. The results could help indicate the efficiency of such an IDS methodology for a scenario where features relevant to detecting malicious behaviour is found.

## 1.4 Research questions

Derived from the discussed issues, the following research questions has been defined for the purpose of being answered and explored upon in this thesis:

1. To what degree does a simulated dataset help indicate the efficiency of ML based IDS in CI?
2. To what degree does use of simulation in developing a dataset benefit the development of IDS in CI?
3. Does a ML model built on a simulated dataset fulfill the requirements of, and benefit detection in the environment of CI?

## 1.5 Planned contribution

For this thesis a methodology of using a simulated dataset for the development of a LSTM classification model for use in IDS has been explored. A LSTM model for classification is built on the behaviour of features under the influence of attack scenarios of varying severity. Evaluation of the capabilities of the classification model could help identify which scenarios of behaviour are hard to detect, indicating what traits the developed model possesses. The scenarios presented are developed to be comparable to measurements made from an ICS environment. The validity of the methodology, in using a simulated dataset to depict the efficiency of a ML model, is also discussed in how it benefits the development of an IDS of CI. The use case will help understand the potential and limitations of the methodology in how it could improve protection of critical and complex systems where acquisition and availability of environmental datasets are difficult to gather or obtain.

## 1.6 Structure

Chapter 2 gives background information and context to the thesis, establishing the knowledge basis of critical infrastructure, machine learning and intrusion detection systems. Chapter 3 describes the process to be used for achieving the described research questions. Here the scope, limitations and scenario are established. Chapter 4 presents the results of the developed dataset and trained model. Chapter 5 discusses the results of the thesis and applies it to the real world to extracts acquired knowledge by interpretation. Chapter 6 describes the concluded knowledge provided by the thesis and gives a verdict of the proposed methodology.

## 1.7 Literature review

There is a lot of previous research in the areas of machine learning, critical infrastructure and intrusion detection systems. To gain insight into the state of the art, this section covers some of what has previously been done, related to the topic of this thesis. Parts of this section is reused from the previous subject, IMT4205, where preparations of the master's thesis begun as part of a project [8]. Since then the material written is still relevant to the thesis and additional papers has been included.

Fountas Panagiotis et al. [9] Explores common intrusion detection methods and what types of attack that are commonly performed towards industries of Critical Infrastructure (CI). Due to the need for expansion and increased efficiency within the industry, there is shown to be a decrease in time and production cost of systems that are implemented. This has however opened up for possible vulnerabilities and attacks towards the critical system, giving the attackers a potential gateway to achieve privileges that should not have been permitted. The paper explores the impact of different types of Denial of Service (DoS) attacks towards a test environment, and explores how accurate different machine learning methods were able to detect malicious behaviour based on the KDD CUP 1999 dataset. Methodologies that learns from the systems behaviour on multiple levels was shown to be the most effective in detecting malicious behaviour.

M. R. Gauthama Raman et al. [2] discuss the role of Industrial Control System (ICS) and Supervisory Control and Data Acquisition (SCADA) which exist within the CI environment. The paper describes how the Industrial Control System (ICS) and the underlying Supervisory Control and Data Acquisition (SCADA) system can be monitored for abnormal behaviour using a Probabilistic Neural Network (PNN). ICS-systems are often found in the industrial sector and critical infrastructure. Therefore a compromised ICS has the potential of dealing large amount of damage leading to the risk of physical damage and loss of human lives [10]. Requirements for the detection system are presented for the detection system to be applicable to the Cyber Physical System (CPS). As the CPS is dynamic, operates in real time and generate data at a high frequency, the anomaly detection system is required to be "fast, reliable, scalable, and sensitive to noisy data..." The PNN was trained on data from real operation, being the Secure Water Treatment (SWaT) located at the Singapore University of Technology and Design (SUTD). Data from the fully functional water treatment plant was collected over eleven days, where the first seven days were under normal operation and the last four days was under the influence of attacks. During the attack, ten different types of attacks were launched by injecting sensory values. The attacks represented previously determined possible attack-points of SWaT being related to flaws in both operation and system. The trained PNN thereby had its smoothening parameter adjusted to a point of liking where the accuracy was at its optimal. The performance of the IDS managed to detect most attacks well, being somewhat challenged by attacks were multiple sensors were targeted.

Andrea Pinto et al. [1] performed a survey on IDS's based on machine learning techniques related to CI. Papers from the last five years was reviewed on how methodology, dataset and results of the created model. The paper is critical to how some of the reviewed research methods does not apply to real world CI as the datasets it is trained on does not represent real-world environments of CI. As discussed, KDD-99 is a widely popular dataset for use in IDS's, however its been more than a decade since its release and therefore does not portray today's cybersecurity area. Though results of a model might seem promising, it might not be the case in the real world, as tests are conducted in much different environment than how a CI network operates. What makes CI especially challenging and different from other environments is a combination of, 1) how it is composed of physical components, 2) utilization real-time transfers of data, 3) the geographical distribution of the environments components, 4) the type of attacker that target CI and the underlying motivation.

The importance of securing CI has also become more immediate due to the industry's transfer over to the fourth industrial revolution [1]. Sensors and components with the purpose of monitoring and controlling the infrastructure are now accessible over the internet. Industrial Internet of Things (IIoT) and 5G communications are being applied in the industry. This makes operation easier, however opens the platform up to a wide spectre of attacks increasing the likelihood of vulnerabilities [11].

In the process of evaluating and comparing use cases and results of IDS based on machine learning [1], it was found to be challenging determining how certain results of implementations differed from each other. Papers utilized a wide variety of different metrics to evaluate the performance of its system. Combined with different types of detection techniques and use of different datasets, comparing the systems become a difficult task as there is no standard for how to determine the best performance amongst different systems. The most used performance measurements of IDS based on ML are; accuracy, precision, recall and F1-score. To better get an understanding of of the ML algorithm; MCC, confusion matrices, specificity, sensitivity and the kappa coefficient are used. To determine Anomaly-based IDS specifically, the "detection rate" and "false alarm rate" are also relevant to the performance of the algorithm. Another factor that will help in determining if a system is capable of detecting intrusions within Critical Infrastructure is the "detection latency". How long it takes for operators to get insight to an attack is essential to the outcome of said attack. The operator should be notified as soon as possible of a potential attack, as time is critical to be able to resolve the issue. The longer an attacker is present on a system, the more potential damage has been caused.

Mohammadreza Begli et al. [12] proposed an architecture that secures the remote healthcare system using an IDS built on the Support Vector Machine (SVM) machine learning algorithm. Due to the differences of requirements and capable processing available it was opted to use a multiagent system. Agents were categorized between found roles within the healthcare system; patient agent, nurse agent, physician agent, ambient agent and databse agent. These agents were again distributed amongst the layered architecture of intrusion detection. This was done as it would encompass how sensors of each of the agents have different likelihood of being targeted by an attack. At the lowest layer, anomaly detection was used to monitor the patient agent and the ambient agent. Given the energy-restrictions of the devices used by the agents and low likelihood of the devices being attacked alone, anomaly detection was chosen as a suitable detection system. Anomaly-based Intrusion Detection System (AIDS) has a low power consumption and computational cost making it fitting for the devices used by the ambient and patient agent. The second layer contains the nurse agent and physician agent. Both agents are less constrained by the access to power and can utilize more computation for its detection method. SIDS was therefore chosen. The top layer containing the database agent utilized a hybrid intrusion detection system to gain the benefits of the high detection rate from AIDS and high accuracy of the SIDS.

R. Vinayakumar et al. [13], develop an IDS built using Deep Neural Network (DNN) as a learning model. The system is able to perform in real-time while being a hybrid between a network and host-based approach to detection. The Host-based Intrusion Detection System (HIDS) functions by monitoring for activity on the host. Most commonly this is implemented by monitoring application files, system calls and operating system. These audit-trails can help identifying and distinguish between known and unknown application as their attributes will generate unique identifiers. The advantage of using HIDS is that monitored data is not obfuscated by encryption as with network traffic, this means that HIDS is able to provide details of the categorised attack. However, to be able to determine the origin of the traffic, a lot of data is needed from different sources within the host. Network-based Intrusion Detection System (NIDS) are prone to a high rate of false positives and false negatives, meaning that it will either fail to detect real attacks or classify normal behaviour wrongly. A proper threshold that determines when an attack is to be classified as such is challenging to set. Both the behaviour of legitimate traffic and how attacks functions are evolving and changing. A self-learning system is therefore implemented into the system as a measurement for the system to be able to change and adapt to the behaviour by continuously learning. The system is thereby introduced to present day attacks and usage.

The authors of [13] also states the importance of feature selection of any model for intrusion detection. The process helps in identifying what features are relevant to achieving a good result. Removing features due to irrelevance or redundancy help reducing the dimensionality that is to be processed while preserving the features that achieves accurate classification. This reduces the amount of data that must be processed, especially as the amount of processing needed escalates exponentially, described as the “curse of dimensionality”. In addition to reducing the training and testing time, feature selection can also improve detection rate as features that might not relate to its classification is removed.

Given the shortcomings and issues of KDDCup98 and KDDCup99, the HIDS was built using ADFA for both Linux (ADFA-LD) and Windows (ADFA-WD). These datasets were created by collecting sequences of system calls and Dynamic Link Library (DLL), of a local system that was open to and under the influence of various attacks. The performance of a DNN is highly dependent on the optimization of its parameters [13]. This was achieved by testing a smaller architecture of the DNN where adjustments to; the number of hidden units, the learning rate, and the activation function was made to find the optimal results. This way changes to the architecture could be made more frequent, while allowing for more frequent iterations and adjustments to find the optimal parameters.

Huan Yang et al. [14] proposes a deep-learning approach to network intrusion detection that incorporates detection of attacks that are aimed towards Supervisory Control and Data Acquisition (SCADA) systems. SCADA systems are widely different from normal networks in how the network operates. SCADA make use of different kinds of protocols which regular IDS created for corporate/campus network would not recognize. Attacks explicitly aimed towards the SCADA system would therefore not be recognized as such. The proposed IDS is therefore built using a dataset that incorporate both “conventional attacks” and “specialized attacks”. The network traffic is monitored by collecting the network packets generated by all SCADA hosts using port mirroring, as this does not interrupt normal operation. As a part of the classification system, one of the classes which the traffic can be appointed to, was the class of “unknown”. Once a large amount (here set to 100) of traffic instances is set to “unknown”, k-means clustering will be performed to distinguish between instances. An operator will then have the option to analyse the clusters and provide them with labels. Once labeled the neural network can be re-trained on the new data. This way the model has the capability to adjust to trends and new types of attacks. Evaluation of the IDS was performed in two sessions. Firstly the IDS was only evaluated using familiar traffic behaviour and attacks. The model achieved a 99.38% accuracy. In the second experiment, the model was tested on unknown attacks. After the process of being re-trained on the new types of attacks the model achieved an accuracy of 99.84%. These results are sufficiently high for use in real-world systems and suited for SCADA systems.

Ángel L. P. Gómez et al. [15] considers the insufficiency in available datasets that enables anomaly detection techniques in ICS. It is here proposed a methodology for how to reliably generate an anomaly detection dataset that can be used for the purpose of deep learning. The methodology consists of four steps; Attack selection, determines what attacks should be launched towards the testbed and included in the dataset. The second step, attacks deployment, describes the process of how and towards what nodes the attacks should be deployed. The third step, traffic capture, describes how data of the deployed attacks should be captured, including both noraml and malicious behaviour. The last step of the process of creating the dataset is features computation. For this step, features are extracted from the captured traffic so that it can take the form of a complete dataset.

Laghrissi F. et al. [16] proposes a deep learning Long Short-Term Memory (LSTM) model for use in IDS. Two comparable models are created with the use of different feature selection techniques. ”Principal component analysis” was shown to perform better than ”Mutual information” when benchmarked on the KDD99 dataset. LSTM was used in a binary and multiclass classification model. Both models were composed of three LSTM-layers which each had a Dropout-layer of frequency 0.1. ”Sigmoid” was used as the activation function together with ”Binary crossentropy” as the loss function in the binary model. For the multiclass model, ”Softmax” was used as the activation function with ”Sparse categorical crossentropy” as the loss function. The model used ”Adam” as the optimizer and was trained on 50 epochs. The proposed model was able to outperform the other compared methods while maintaining a low amount of features.

# 2 Background

This chapter goes through some necessary background to get some context to the domin of Critical Infrastructure (CI), machine learning and intrusion detection. Section 2.1 describes what CI is and what makes Industrial Control System (ICS) different from other types of environments. The challenges and requirements of CI is also described to understand what malicious actors and attack vectors the area must be protected against. Section 2.2 gives an introduction to the basics of Machine Learning (ML) and neural networks which is used in this thesis. Explainable artificial intelligence (XAI) is also described for the purpose of attempting to understand and gather knowledge from ML. Section 2.3 gives an introduction to how Intrusion Detection System (IDS) work to mitigate and prevent cyber incidents.

## 2.1 Critical infrastructure

Critical Infrastructure are systems and assets that are essential for a society to function [1]. Energy [17], water-treatment [2] and healthcare [12] systems are amongst crucial components that if disrupted would have profound consequences. Such events could affect national security and impact human needs, safety and livelihood, potentially leading a nation to a state of political and economical crisis. As the presence of CI is a requirement for other systems and services, losses in CI will impact a wide variety of necessary applications, acting as a domino-effect. Losses in areas such as network-communication and energy generation or delivery would restrain access to network and power. Destruction within one CI-sector also has the capability to affect other areas of CI and therefore has the potential to impact a nation to a greater extent [9].

The consequences of failures and the importance of uptime in CI, is well known to malicious actors. Attempts and successes in infiltrating the environment could give motivated attackers the capability to manipulate and sabotage systems that control and manages the infrastructure [18]. In the setting of a power-grid, this could mean manipulating critical power functions and possibly shutting down the grid causing blackouts and loss in function of appliances with no backup power.

In section 2.1.1 the environment of Industrial Control System (ICS) will be introduced as it is often found in operational systems, such as in CI. Thereafter innovations and challenges of the environment is presented in section 2.1.3 and 2.1.4 respectively. Section 2.1.5 describes and discuss malicious actors capabilities and motivation in relation to CI. Further the ICS's role in IoT botnet's and attack methodologies in CI is described in section 2.1.6 and 2.1.7.

### 2.1.1 Critical infrastructure environment

The environment of CI differs from regular network traffic of other areas. Given the operational needs and the components that is placed in the environment, traffic behaves differently. The environment can be described with the Industrial Control System (ICS) architecture. "Industrial Control System (ICS)'s are composed of the interconnection of different computers, electrical and mechanical devices used to manage physical processes" [19]. They are often very complex systems that contain sensors, actuators that are controlled and monitored. ICS's manages normal network traffic as well as physical and low-level processes. The ICS Purdue model architecture is depicted in figure 1. The architecture can be separated into three zones; Enterprise, Control and Safety Zone each separated into level of operation.

1. **Enterprise Zone**
   The enterprise zone contains IT devices found in most commercial organization. The zone can be segmented into level 4 and 5. Level 5 is the enterprise network, which is usually connected to the internet for usage by the office staff [20]. Level 4 describes Business logistic systems which are IT devices that supports the operation on the enterprise network, such as printers and mail exchange servers.

2. **Control Zone**

The control zone is separated from the enterprise zone with a Demilitarized Zone (DMZ) that secures traffic between the IT operations found in the upper zone and OT operations of the devices in the control zone. The zone contains four levels of devices.
Level 0 includes devices directly connected to the physical processes, such as sensors and actuators. A sensor could for instance be used to measure temperatures while an actuator could be responsible for opening or closing a vent.
Level 1 describes intelligent devices that control the devices in level 0, based on their function. Such devices are Programmable Logic Controller (PLC) which can execute orders to actuators based on input from sensors and controllers. Intelligent Electronic Device (IED)'s, function similarly to PLC's in being capable of performing operations, but to a greater extent. They are capable of more complex tasks such as detecting faults, local and remote control. The devices are often created to be able to sustain harsh conditions and maintain operation for more than 10-15 years [19]. In CI and ICS environments it is important that the devices are capable of real-time operation with low levels of jitter and delay [1].
Level 2 devices of the control zone are control systems that are made to manage automated processes and operate the devices of level 1. The Human Machine Interface (HMI) is one such component that is used to manage values, alarms and observe trends [19]. Supervisory Control and Data Acquisition (SCADA) devices are also found here and is used to collect data from different fields, so that it can be centralized and analysed.
Level 3 are devices which is connected to the enterprise zone through DMZ. The Engineering workstation is a platform which hosts software for controlling and managing controllers found on level 2.

3. **Safety Zone**
   The safety zone is a designated zone for devices and systems that ensures the ICS operates in a secure state. Through monitoring, the Safety Instrumentation Systems (SIS) can flag unsafe operating states, restore the system back to a safe state or shut off areas of operation for analysis and prevent damages [20].

### 2.1.2 Critical infrastructure requirements

For operation of CI to be stable, communication between the components must have low levels of jitter and delay. Due to the devices and how they must operate to support the environment, traffic that is generated is significantly different from other types of network traffic. Emphasis is put on the availability of data transmitted and access to devices as control over the physical system is essential to prevent damages of components and maintain production [1]. Protocols in which supports fault tolerance and reliability is therefore prioritized. This however comes at a cost of the security implemented into the devices and protocols that are used [19]. Security features such as authentication and encryption has not been prioritized as they are mainly created to operate in closed environments [3]. Though implementing security is not mutually exclusive to the availability of the traffic it could impact the system by introducing some delay.

As CI-environments develop and produce essential resources, continuous production is a requirement of the systems. This could make management tasks and service of the system hard as it can not be interrupted for long periods of time [1]. The CI-environment is composed of a wide variety of components specifically built for the type of service to be fulfilled. Traffic and data flowing through the network is therefore heavily dependent on those systems. Traffic of different CI-sectors also differs from each other as there are a wide variety of protocols used to handle ICS's.

### 2.1.3 Industry 4.0

The fourth industrial revolution also called "industry 4.0", describes the innovations which has been taken place in the area of manufacturing. The evolution of technical capabilities over the past decade has enhanced the whole value chain of the life cycle of manufactured products [21]. The industry 4.0 environment are composed of interconnected computers which communicate and interact with each other without human intervention. The use of AI, smart sensors, cloud and big

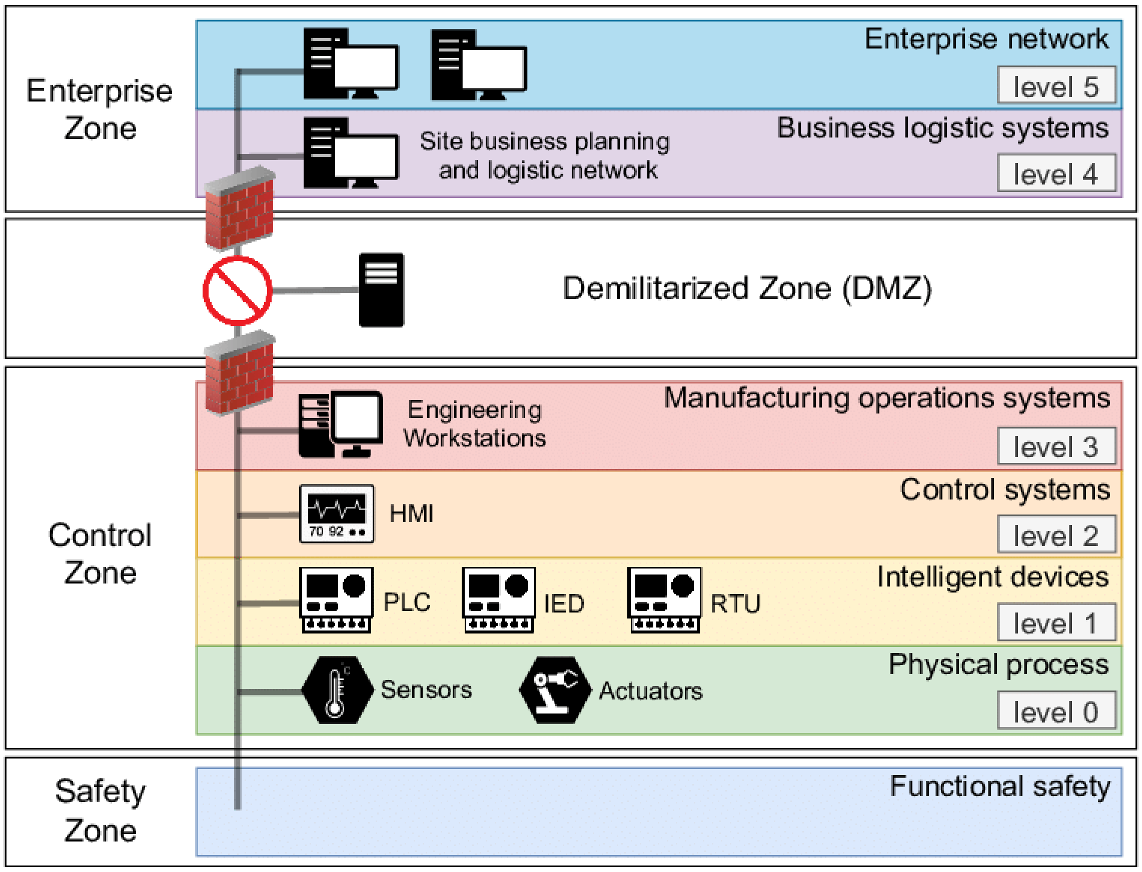


Figure 1: ICS Purdue model architecture [19].

data analytics has enabled automation and increased efficiency of production. The components of the interconnected production environment are not operating independently from each other. Based the context which one component receive from another, it can make decisions and coordinate with other parts of the operation. Automation and smart manufacturing allows for optimization of the production line, making improvements to waste reduction. The integration of a dynamic cyber-physical control system allows for improvements to the reliability and the flexibility of the operational environment. Virtualization of the operational environment through the use of digital twin, processes can be simulated. This can enhance the agility and adaptability of the environment further enabling discovery for areas of improvement. Optimizations to manufacturing is important to improve on the sustainability in production. Reducing waste and increasing resource efficiency means a smaller economical and environmental footprint. Though manufacturing is the most notable area where Industry 4.0 has taken place, there is a wide area of different applications. As mentioned industry 4.0 follows the products created throughout the whole life cycle. Industry 4.0 is also prevalent in purchase and supply management, design, evaluation, distribution, prototyping, quality control and more [22].

A large component of industry 4.0 is the incorporation of "Internet of Things (IoT)" devices. IoT devices allows for *human-to-thing(s)* or *thing(s)-to-thing(s)* communication to provide a service [23]. What makes IoT devices particularly different from other types of computers is that they are often heavily resource restricted. To allow for low cost and small form factors the devices have limited computational power and battery. This affects the devices ability to perform security-related tasks. As the devices are low-powered, computationally and resource constrained, it is limited how effective authentication and encryption can be implemented [24].

### 2.1.4 Critical infrastructure challenges

The CI environment can grow to be highly complex, built from a large amount of nodes that communicate and cooperate. Managing and controlling the operation is costly, which is why technology from the 4th industrial revolution has been incorporated into production [1]. This has supported the production, making it more cost effective, efficient, sustainable and automated [4]. However, incorporation of such technology has expanded the attack-surface, opening the environment up for more types of attack vectors, exploits and vulnerabilities [9] of greater complexity and diversity [1]. Previously, components has been protected as they have operated on proprietary hardware on isolated networks with no connection to the internet [5]. As remote access and monitoring has been implemented with the ICS environment, potential doors has been opened to give access and control to malicious actors. An environment of interconnected IIoT devices coming from different manufacturers with varying implementations of authentication, security, technologies, protocols and exposed ports makes managing security significantly harder.

Protocols developed for ICS devices can be grouped into two categories based on who it is that develop the system [3]. The protocol is either developed by industry standard organizations, where vendors of different organizations can implement the protocol to work with other vendors. The second type are protocols developed by the devices manufacturer. This type of protocol is designed for their own hardware and the previously discussed industrial needs of ICS. The lack of built-in security features makes the device not fit for availability from the internet. Despite this, a large amount of ICS devices are found exposed on the internet [3]. With few to none security features, attackers can remotely access the devices to control hardware, monitor the behaviour and perform Man-in-the-Middle (MITM) attacks.

### 2.1.5 Malicious actors

Given the differences between most networks and the infrastructure found in CI, attacks aimed towards the ICS require a tailored approach. This was especially true when the industrial components was isolated. However system dependencies between OT and IT has proven to be one possible attack method for adversaries, which even novel actors without deep knowledge of ICS systems could perform [25]. Malicious actors which targets CI could therefore come from any level of skill and resources. Most notably are the adversaries with the most available resources available to them.

Advanced Persistent Threat (APT) -actors have a large amount of resources, experience and knowledge giving them the capabilities to perform highly sophisticated attacks. APT's are hacking groups that are financially supported by their government or companies [26]. Attacks conducted by APT's can be characterized by their methodology. The attacks can be conducted over a long time, APT's ensure stealthiness by conducting the operation methodically. The motivation of the attack is often highly specific, based on a clear goal. Burita and Le [26] divides APT's into four types;

1. **Nation-state actors**
   The nation-state actors are politically motivated hackers funded by their government and can be used for military purposes. This entails activities such as disrupting strategical points of interest, critical infrastructure being one potential target.

2. **Organized crime actors**
   Differently from nation-state actors, actors of organized crime are more often financially motivated. Financial gain could be achieved by targeting commercial businesses with ransomware, blackmailing or trading industry secrets.

3. **Espionage actors**
   Espionage actors are interested in obtaining confidential information. This could be for both financial and political interest. Attacks with the purpose of espionage are especially stealthy to maintain a position where information of high confidence could be picked up.

4. **Cyber terrorists**
   Cyber terrorists, similarly to terrorists are actors seeking to cause harm and destruction, now within the digital realm. Cyberattacks conducted for the purpose of terrorism are conducted to affect and influence the public. The motivation for such an attack could be political, ideological or religious.

As APT's have access to large amount of resources it is likely that they have the capabilities to develop or access to zero-day attacks. A zero-days attack is a cyberattack which utilizes a vulnerability or exploit which has not been publicly disclosed [27]. The "zero", references the amount of time the method of attack has been known for; in other words, its completely new. Because the vulnerability or exploit is not known, zero-days are particularly hard to defend against and give the adversaries an advantage in successfully conducting the attack. Access to zero-days, large amount of resources and heightened motivation enables the APT to develop tailored attacks specific to an organization. Given such a scenario it is not likely that such an attack would be identified, nor prevented.

### 2.1.6 IoT botnet

Botnets have shown to be one of the most prominent threats to system and IoT security with the capacity to cause large amounts of damage through Distributed Denial of Service (DDoS) attacks, theft and spam [28]. A botnet is a large collection of infected devices which are taken control of by one Command and Control (C&C) server. All the infected devices (bots), are thereby under control of the server, awaiting orders. The collection of bots can thereby be used to send massive amounts of packages towards a point of interest, overflowing it with bogus traffic. Genuine traffic is therefore discarded as the service does not have the capabilities of handling all of the requests. As the traffic comes from many different sources, blocking said traffic becomes hard as it could be composed of thousands of different IP addresses of multiple ranges. The consequential downtime of the services could have significant impact, dependent on the service which it provides.

The Mirai botnet is one of the largest botnets and has been the cause for large scale DDoS attacks, which hindered access to significant portions of the internet [29]. The Mirai botnet expands on to vulnerable devices which are constantly being scanned for. Particular services and ports known to host vulnerable services are targeted and attempted exploited and brute forced. Many impacted devices are IoT devices, due to their limited computational power and weak implementation of security [30]. Security in IoT devices has not been of priority as they are made to be cheap, lightweight and easy to manage and use. Variants of the Mirai botnet has over time expanded its pool of services and ports which are targeted. From mostly exploiting Telnet, the actors has also included exploitation's of SSH, HTTP and other TCP-based ports to their toolbelt. Malicious actors are financially motivated to expand the capabilities of its botnet to provide the infrastructure as a service on a marketplace. Here non-sophisticated actors are given the capability to conduct large-scale DDoS attacks by renting availability to the infrastructure as a service [31].

Environments composed of large amounts of IoT devices could be especially subject to automated attacks coming from botnets. The actor has an incentive gaining control over the environment as it would make a large addition to the botnet, expanding the potential size of attacks to be conducted. Devices of the IoT environment is also harder to manage given the scale and complexity. A large collection of different types of devices, technology, manufacturers and protocols, increases the risk of having a vulnerable component [32]. Such a vulnerable device could function as a point of entry to the infrastructure. Having the environment as part of a botnet means that it could be partaking in DDoS campaigns, without the knowledge of the owner or administrator. Disruptions could also be made towards the host environment of the IoT devices, given the control which the *botmaster* has over operational pieces. The botmaster could also be given access to confidential information wich is transmitted over the network.

### 2.1.7 Cyberattacks in critical infrastructure

APT activity has been identified in attacks tailored to ICS system and components. Stuxnet was a worm which targeted SCADA systems. The malware was highly sophisticated, utilizing four zero-day vulnerabilities to achieve its goal [33]. Assessments of experts conclude that the developers of Stuxnet had extensive insight into the programs, computers and systems of the Iranian nuclear plant and uranium enrichment site. This made testing and development of the malware possible, ensuring its impact on the production.
The Industroyer2 was a malware targeting the Ukrainian power grid. It is a variant of a previously used malware which targeted ICS systems. The most recent variant specifically targeted the real-time communication protocol used to monitoring and control [34]. As a consequence of the attack damages has been done to physical infrastructure, impacting the power distribution and causing blackouts.
The given attacks are examples of highly sophisticated and tailored attacks targeting the ICS causing significant consequences. What makes these attacks different from attacks targeting commercial networks is that they are made to handle complex, interconnected components and OT devices. This could for instance be exploiting protocols with no implemented security to spoof or manipulate packages [35].
Attacks targeting ICS can be separated into two different categories based on their attack vector; *Network-based attacks* and *physical-based attacks* [19].

1. **Network-based attacks**
   Network-based attacks targets packets, protocols and policies that are in transit between the components of a network. From the network an attacker will often attempt to know what devices are currently located on the network.
   Reconnaissance is performed to map out the layout of the network, this can be done both passively and actively [36]. *Passive reconnaissance* is performed by laying still on the network, capturing packets that are passing by. Any communication that is not encrypted, an attacker could potentially read to gain insight to what components of the network are reachable. *Active reconnaissance* is used by an attacker to gather more precise information. Over the network the attacker can send out multiple packets over a range of IP-addresses on multiple ports and see what services are responding.
   *Man-in-the-Middle (MITM) attacks* enables an attacker to sit in between two parties and read or modify what they are sending to each other. This could be achieved by the attacker by *spoofing* his own address, portraying himself as the receiver to the responder. With the knowledge of what packets are being sent, the attacker could be capable of performing *replay attacks.* Re-sending previously sent packets known to be valid, with adjustment to the sequence number could cause interruptions in the operation. Spoofing also enables an attacker to perform *packet injection.* If the attacker portrays as a controlling station, packets could be sent to actuators, causing them to perform unwanted tasks and cause malfunction in the operation [37]. As ICS protocols have limited security implementations, the protocol might be used to perform Denial of Service (DoS) attacks. This could typically be if protocols can be used to create an increasing amount of packages filling the network, hindering ordinary packets in reaching their destination. DoS can also be achieved by spoofing, by constantly misguiding traffic so that the target is never reached [37].

2. **Physical-based attacks**
   Physical-based attacks are targeting internal cyber-physical systems. The physical components targeted are field devices such as sensors and actuators that could be located on geographically remote sites. Attacks targeting the physical systems are achievable only after a foothold on the network, usually accomplished performing a network-based attack [19].
   A *stealth attack* is performed by making small alterations to values which the field devices report. As the small changes are challenging to detect the inaccuracy could persist over an extended period of time, which in return can deteriorate and damage the components. Similarly *Direct damage attacks* aims to bring production to an unsafe state by manipulating values which determines the state of an operation [38]. This could for instance be adjusting the measured temperature to something lower, forcing production to operate under temperatures it is not fit for.

## 2.2 Machine learning

Machine Learning (ML) is a branch of artificial intelligence where gathering and collection of knowledge is simulated by use of algorithms [39]. ML has become a primary methodology for handling large amounts of data through extracting knowledge by statistical measures and computational power. With the help of ML logic can be gathered from large amounts of data that would otherwise be incomprehensible to humans. Through the usage of algorithms that handle the data, it is able to learn from experience without human intervention. The growth of the field comes from newly developed technology providing greater computational capabilities with the capacity to perform demanding tasks more efficiently. ML creates results in forms of rules, functions, relations distributions and more to represent knowledge as a model that can be used to support underlying processes [40]. This can be used in a wide variety of tasks to help in decision making and prediction. ML has emerged as a useful tool in areas of medicine, finance and retail, with the potential to revolutionize the industries [41]. As the need to handle an exponentially increasing amount of data, ML will continue to be an essential tool in performing research, increase efficiency, quality and the capabilities of information handling and knowledge gathering.

### 2.2.1 Machine learning approaches

Machine learning can be approached utilizing one of three methodologies.

1. **Supervised learning** refers to a ML-model where data utilized for learning is labeled. This means that the training is based on the previous knowledge of what type of data is inserted into the model [39]. Using only the featured values of a datapoint, the ML-model will learn by attempting to predict the label of the datapoint, adjusting its weights based on the result. A trained model can then be used to classify unseen data based on the trained rules that has been generated.

2. **Unsupervised learning** is used to generate logical connections between datapoints that are not labeled. Based on given input, characteristics of the data are found from the attributes. Unsupervised learning can be described as self-organizing or adaptive learning algorithm as there are no exterior factor that administrate the correctness of what is being learned [42]. This is mainly used for clustering and associations algorithms where observations can be separated and grouped accordingly to their measured features.

   A hybrid of supervised learning and unsupervised learning can also be used to achieve a greater performance. This combines the experience gathered from supervised data to predict the outcome of unknown data [43].

3. **Reinforcement learning** is based on giving feedback to the algorithm based on actions that are made. Through trial and error the machine can find optimal actions for given scenarios. Based on previous knowledge of when it was rewarded for its action and when it was punished, the model make decisions to optimize its score. Instead of providing the algorithm with information of what class of given datapoints belong to, the algorithm is rewarded accordingly to right decisions made [42]. It will thereby know when it is on the right track, and can explore further alternatives in which will be reinforced or discarded.

### 2.2.2 Machine learning classification algorithms

An introduction to some frequently used, traditional classification algorithms is given as they will be used for comparison to the proposed AI model. Comparing the results it can be observed how the methods distinguish themselves from each other to find advantages and disadvantages.

- **k-Nearest Neighbors (KNN)**
  KNN is used for classification in supervised learning. It is one of the most simplest algorithms

for classification. The class of an observation is determined by the majority class of the $k$-numbers of the observations which are the closest distance to the sample. The distance is determined by the euclidean distance of all the features

- **Decision trees**
  A decision trees make prediction of what class a sample belongs to based on feature values which distinguish the classes from each other. Based on the value of a feature, a path is chosen along another branch of the tree. The branch might lead to another feature or a predicted class. Classification by the use of a decision tree can be expanded upon by the use of *random forest*. Instead of using a single decision tree, the random forest relies on various decision trees [44]. Each tree makes a prediction of a sample following the decision nodes of the tree, until the leaf nodes are reached. From there the majority class which was predicted is determined as the final prediction.

- **Support Vector Machine (SVM)**
  SVM is used for both classification and regression in supervised learning. The idea of the SVM is that it attempts to find a hyperplane of a higher dimensional space, where datapoints of different classes are optimally separated [45]. The orientation of the hyperplane is found by the use of two support vectors, positioned parallel on each side of the hyperplane. The distance between the hyperplane and the support vector (using hard margin) is the same as the hyperplanes distance to the closest datapoint. The optimal hyperplane has therefore been found where the distance between the two support vectors are the furthest from each other. The farther apart the support vectors are from each other, the better the model's ability to generalize the data.

### 2.2.3 Artificial neural network and Deep learning

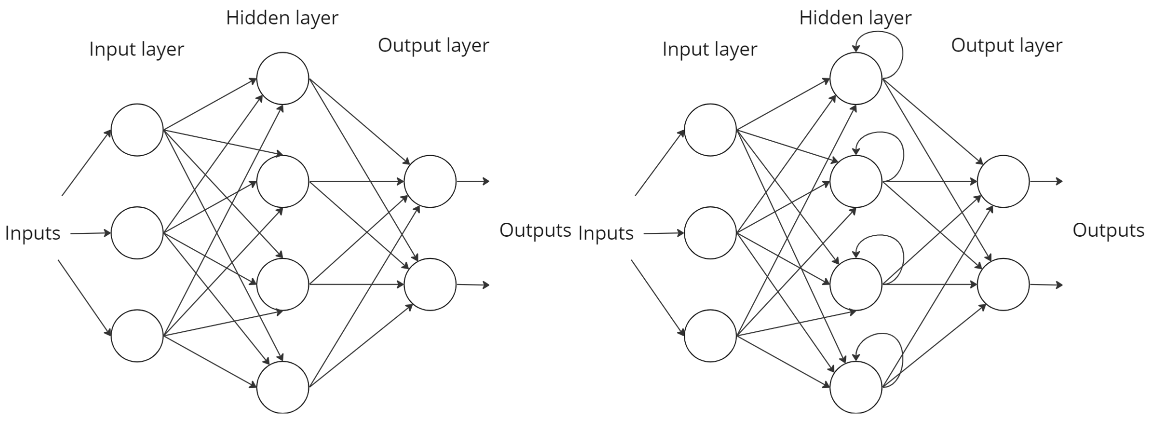
An Artificial Neural Network (ANN) is an approach to machine learning which is based on the functionalities of the biological neural network [46]. A neuron of the network takes an input data where it is processed before outputting an additional value. Initially features of the model are used as an input to the neuron and multiplied with an adjusted weight. The weight signals the importance and influence of that particular feature and how much it should influence the decision-making. Each weighted feature is thereafter summed with an additional bias for the model to explore other areas of optimization. The sum is passed through an activation function that defines the property of the neuron. The function determines the format and value of the output, based on the summed and weighted input. One such instance of a neuron is also referred to as a *perceptron*. The Artificial Neural Network takes shape when multiple perceptrons are combined and interconnected in multiple layers. The first layer of a neural network is referred to as the *input layer* while the last layer of the network is the *output layer*. Any layers in between the input and output layer is referred to as a hidden layer. An ANN composed of more than one hidden layer is known as a Deep Neural Network (DNN). An ANN is trained through iterations of making predictions, determining accuracy and adjusting the weights of the features to optimize the performance of the network. Weights of the perception input are adjusted based on the results of predictions made, which is referred to as *back propagation* [47]. For each iteration the error rate is measured and used to determine the direction weights should be adjusted. Once the error rate achieves an acceptable performance, the training is complete.

An ANN's features and capabilities is heavily dependent on the network topology. Most commonly found network topologies are *Feed-forward Neural Network (FNN)* and *Recurrent Neural Network (RNN)*.

**Feed-forward Neural Network (FNN)**
In a FNN the hidden and output layers take the ouputs of the previous layers as an input. The information obtained in the input layer only flows forward to the next layers. The information is thereby not used to influence the previous or current neuron.

**Recurrent Neural Network (RNN)**
In a RNN outputs of a perceptron can be directed towards previous, current and the next layer of the network. This gives the network the benefit of simulating a current state of the network,

where its behaviour can change over time [46]. By including the previous outputs of perceptrons into the current prediction, the network posses a “memory” of previous inputs. This enables the network to process sequential inputs to make predictions. RNN can therefore process data that is dependent on the results of the previous sample for its analysis [48].



(a) A model of a Feed-forward Neural Network (FNN) with three layers.

(b) A model of a Recurrent Neural Network (RNN) with three layers.

Figure 2: Examples of Feed-forward Neural Network (FNN) and Recurrent Neural Network (RNN)

RNN has since its origin been heavily used for a wide variety of applications using sequential data [49]. The results of RNN in domains such as language models and speech-to-text has shown to be especially positive. One limitation of the RNN is the concept of *vanishing gradient* and *exploding gradient*. In use of a simple activation function, for each layer the loss function is introduced to, the gradient is exponentially reduced or increased, approaching zero or infinite [50]. Long Short-Term Memory (LSTM) was created to address the problems. While training, it is determined whether information in the network is useful. The information which is maintained is compared to incoming data. The model is capable of deciding what information should be overlooked and what should be remembered [51]. In comparison to a simple RNN architecture, LSTM can remember and utilize data for longer periods. Instead of having a single tanh for each layer, an LSTM is composed of three different “gates” [16].

1. **The forget gate**
   Both the new input data and the previously kept data is taken as input which is used to decides what information to delete using a sigmoid function.

2. **The Input/Update gate** This partition decides what new information should be stored. The state of the cell is updated by combining; a sigmoid layer which decides what values should be updater, and a tanh activation function which provides candidates to be added.

3. **The Output Gate**
   The state of the cell as well as the new input is provided to the output gate which decides what is going to be the output. The cell state is passed through the tanh layer to gain values between -1 and 1. The new input is passed through the sigmoid function which decides which of the resulting tanh values is output.

Figure 3 displays the LSTM module and its gates as a whole.

#### 2.2.4 Stateful and stateless Recurrent Neural Networks

RNN’s and in extension LSTM’s, are especially capable of processing sequential data due to its ability to retain the information of previously processed samples and use it for its decision making of the current sample. This has been used for processing data which involves sequential dependencies, for example speech recognition or weather forecasting. In the process of training a model it is most

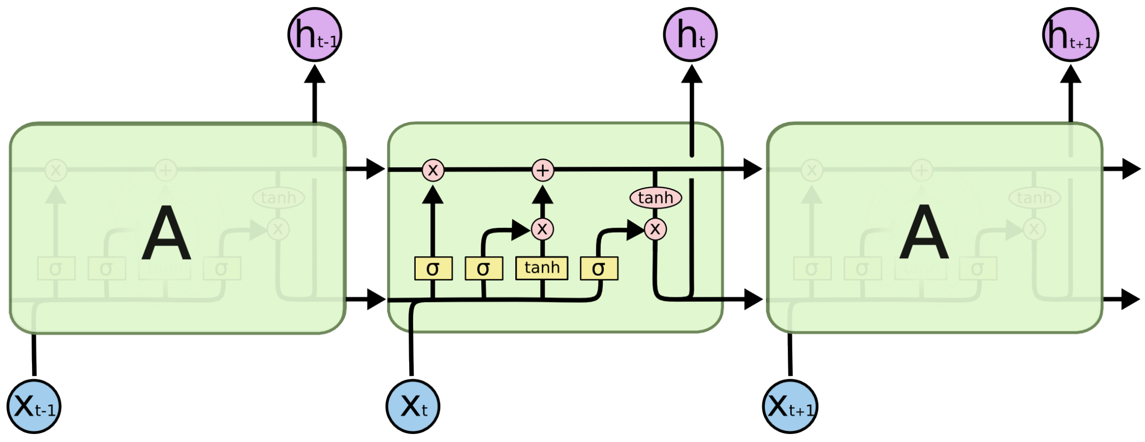


Figure 3: Module of repeating LSTM architecture [52].

commonly trained on batches separately, where each input sequence is processed independently [53]. This is the *stateless* approach to training a RNN. In different to a stateless RNN, the *stateful* RNN preserves its cell state through batches instead of discarding it [54]. As the state is not reset for each batch, the previously processed samples are capable of determining the current sample. This is useful for when dealing with sequential data which retain long-term dependencies.

#### 2.2.5 Feature selection

To be able to extract knowledge from the data which is collected using a ML model, the data which is fed to the model should be optimized for its purpose. [13] discusses the importance of feature selection in identifying various types of attacks. Feature selection is used to remove features which are irrelevant or redundant to the identification of attacks. This can help improve the accuracy of the model and computational cost of training. Data which does not give any indication of incoming attacks should be removed as the inclusion of such features could confuse or give false indications to the model under training. Inclusion of features which gives no indications also increases training and testing time, as a whole other dimension of data must be processed. This also goes for redundant data. Even though the feature could indicate an attack, if it is redundant to another datapoint which portrays the same indications it does not provide any new information to the model. The feature should be removed as the training time of the model increases exponentially, hence the *curse of dimensionality.*

An increase in dimensions has the disadvantage of making the data more sparse. This makes classification tasks much harder as data is distributed amongst more 'axis' [55]. The sparsity hinders the model in finding strong statistical relations in the data. Too many features also makes trained models highly specific to that particular dataset, thereby being overfitted. Rather than providing a general model to the desired problem description, the model portrays the specific datapoints. It is in [56] argued that feature selection can both increase system performance and the classification accuracy of IDS's. A large number of dimensions could especially impact the performance of ML algorithms in real-time environments [16]. Feature selection is performed by measuring the relevance of a feature and maximizing the classification accuracy based on a feature subset [57].

#### 2.2.6 Machine learning datasets depicting critical infrastructure

For a ML model to make accurate predictions of an outcome, the basis of the model must be based on the real environment it is attempting to predict. In the case of using ML to detect malicious behaviour, the model should have been built on data that reflect the behaviour accurately, both under normal and abnormal circumstances. As the behaviour of a network changes amongst different industries, configurations, protocols and usages, a model which is built on traffic from one environment, is not necessarily applicable to another environment.

As discussed previously the behaviour of CI stands out from other types of traffic. Therefore, any attempt to use ML for intrusion detection should be based on this particular infrastructure [14]. Unfortunately, there is a lack of available datasets which portrays CI [1], [4], [19], [48]. Presented models are found largely to be based on outdated datasets. Most used dataset for IDS testing is KDD-99 which originated in 1990. Results of the evaluation made does therefore not portray how the detection system would perform in real world environment. CI in particular behaves differently, this also goes for the attacks which are aimed towards CI. CI involves different types of attacks as the system is built on other types and highly specific components. Attacks relevant to the system should therefore also be introduced to the ML model.

The lack of datasets which portray the behaviour of CI also comes from the reluctance of organizations to share the data. The data which must be provided to create a dataset, could entail sensitive information about the system. From this, vulnerabilities in the environment could be discovered by malicious actors, giving them material to develop a means of entry. Though the process of developing highly targeted attacks based on these types of leaks is very complex, as discussed in section 2.1.5, the actors which has the capabilities to do so are also motivated to target CI. In developing a dataset reflecting the behaviour of a CI environment, introducing the environment to malicious activity would also not be possible for collection of relevant data, as this might lead to actual harm such as damaging or destroying components.

### 2.2.7 Explainable artificial intelligence (XAI)

In working with Machine Learning and especially Deep Neural Network, it can be unclear exactly how developed models function. ML models through training are capable of finding complex relations and logic. Though such a model is able to accurately classify and make predictions, the inner functions of such a model is *opaque* to those who use it. DNN's are essentially a black box. Explainable artificial intelligence (XAI) encompass techniques which aim to make AI more comprehensible for users to understand how results of the model came to be [58].

In working with ML models, it is important to know how it functions even though it performs well. It is in [59] provided an example of how an image classification model misinterpreted the difference between a wolf and a husky. What was found with XAI, was that the model saw that pictures of wolves was more likely to be taken in environments with snow. Instead of classifying based on the look of the animal, the model found that taking the environment into consideration was an efficient determinant. Though accurate, the classification model was right for the wrong reason. Similar scenarios are possible when working with intrusion detection. XAI can help identify biases and improve the overall performance [60]. An understanding of the ML model, can also provide knowledge in how detection is made. Through analysis of the model it can be determined what exact features are important for the decision making and figuring out what particular scenarios indicate attacks.

DNN's are found to be of benefit for higher accuracy in many areas of ML, however has come at the cost of transparency [61]. Simpler ML techniques in comparison, are easier to understand and analyse, providing a clearer answer to how results were achieved. In attempting to provide transparency to the opaque models there are XAI techniques which could be implemented at different stages of the model development [58].

1. **Pre-modeling**
   Pre-modeling are statistical measurements which is done to the dataset of the model, to provide knowledge of how that data is distributed. This method might help identifying biases in the dataset such as imbalances and missing data.

2. **Interpretable model**
   An interpretable model is a ML model which is created for the purpose of having it provide explanations to its decision making. Such a model must therefore be considered when designing the model.

3. **Post-modeling**
   Post-modeling are techniques attempting to provide an explanation of a black-box model. This can be done through visualization, where areas of importance to the decision making are highlighted by the model.

## 2.3 Intrusion Detection System (IDS)

Networks are handling an increasingly amount of data and devices. Yet, to achieve a secure state of the network malicious packages must be detected and prevented. An Intrusion Detection System (IDS) aims to find the suspicious behaviour and issue an alert such that the event can be handled as soon as possible. The automation helps administrators manage the amount of traffic, however, it is not a given that the IDS will be able to find every malicious package. Intrusion detection is a constant development following the behaviour, tools, techniques and vulnerabilities attackers use to infiltrate a system.

IDS's can be separated into categories based on the method of intrusion detection; Signature-based Intrusion Detection System (SIDS), Anomaly-based Intrusion Detection System (AIDS) and Stateful Protocol Analysis (SPA) as well as two categories based on the localization of the detection mechanism; Network-based Intrusion Detection System (NIDS) and Host-based Intrusion Detection System (HIDS).

### 2.3.1 Signature-based Intrusion Detection System (SIDS)

Signature-based Intrusion Detection System (SIDS) use its knowledge of existing methodologies and patterns in detecting potential attempts of intrusion. A collection of signatures from previous attacks, as well as packets indicating suspicious behaviour, is compared to incoming traffic. A match between the packet and an entry from the list issues an alert of the packet [62]. Though this method of detecting attempts of intrusion is highly accurate and efficient, it does not have the capabilities of detecting new types or modified attacks as these are not a part of the database. This database must continuously be maintained and updated with new forms and variations of existing attacks, as they are disclosed. SIDS is also known as *misuse detection* as it defines general actions which are prohibited by the policy.

### 2.3.2 Anomaly-based Intrusion Detection System (AIDS)

Anomaly-based Intrusion Detection System (AIDS) determine its decision by comparing the current behaviour to a set basis that describes the normal behaviour. The normal behaviour of the system is referred to as a profile, which describes the expected behaviour [63]. If the current traffic is different from the profile by a set threshold an alert is made. Profiles can be made specific to different environments and areas of an infrastructure, making it highly detailed to how different networks and applications operate. AIDS has the advantage of potentially detecting unknown forms of attacks, as such an attack would deviate from expected behaviour. The drawback is that AIDS does not achieve the same rate of accuracy as SIDS. It can in AIDS be hard to define the threshold which determines normal behaviour. Either the threshold is too vague, increasing the risk of not detecting misuse. Otherwise it might be too strict, issuing too many alerts for irrelevant variations.

### 2.3.3 Stateful Protocol Analysis (SPA)

Stateful Protocol Analysis (SPA) also referred to as specification-based detection, functions by tracing the protocol states [64]. The detection method differs from AIDS, by having vendor-developed profiles which are specific to protocols, rather than profiles specific to network or host.

### 2.3.4 Network-based Intrusion Detection System (NIDS)

The location where the IDS is placed can determine what types of attacks it will be able to detect, how early the attack is detected and what data can be gathered about the attack. Network-based Intrusion Detection System (NIDS) is deployed on the network, which aims to protect all of the devices located on the network [63]. NIDS inspect packages that are transferred between hosts and evaluate their contents for suspicious material. In a signature-based approach, the contents can be compared to known signatures, IP-addresses, protocols and more. In anomaly-based detection the traffic-flow, could for instance determine if the network operates as usual by analysing what IP-addresses are communicating with each other.

### 2.3.5 Host-based Intrusion Detection System (HIDS)

Host-based Intrusion Detection System (HIDS) aims to protect the particular machine in which it is deployed on. Since adversaries possess tools which are able to gather system specific information of hosts, vulnerabilities found can be used to gain unauthorized access [13]. HIDS monitor traffic and activity on the host itself by analysing application files, system calls and the operating system. In comparison to NIDS, HIDS can provide detailed information about the suspicious activity it detects, as host contain detailed logs of host activity. Not only do you see the consequence of an activity as in the network packet, but you can see the particular activity that was performed. The downside to having the IDS located on the host is that it uses the computational resources of the host, ultimately degrading its performance [63].

### 2.3.6 Machine learning in Intrusion Detection System

Machine learning as described in section 2.2, covers what ML approaches can be used in IDS. Use of ML and deep learning is seen as a potential possibility of achieving greater performance in ids by using big data [65]. There are proposed many ML and Deep Learning (DL) algorithms for use in IDS. The different approaches has weaknesses and strengths and must be fit for the environment and the issue which it attempts solve. The use of ML does however have some particular challenges when it comes to the use case of IDS [43]. Any common network has large varieties in packets given any behaviour. The variety and diversity in traffic makes it significantly harder for a ML model to detect what particular features of the network makes behaviour abnormal. Similarly, behaviour which is considered normal might change for each established session. The usage of each session might differ depending on the purpose of the established session.

Use of ML still holds a high rate of errors. A high rate of false positives in an IDS results in a large amount of noise for any operational center which has to analyse any issued alerts. This results in the usage of a large amount of resources used on benign alerts which might even make finding the real events hard to find. Tuning the ids to not issuing as many alerts might on the other hand increase the amount of false negatives which is significantly more severe in that there are not given alerts on real attacks.

# 3 The proposed deep learning IDS mechanism

This chapter describes the course of action that is planned to be able to answer the research questions presented in 1.4. Section 3.1 and 3.2 defines the scope and limitations of the thesis, narrowing the field to a viable scale, specifying the area of research. Section 3.3 presents an overview over the methodology, showcasing the planned course of action. Section 3.4 describes the development of the dataset that is to be used for training of the machine learning decision-making model. Creation of the machine learning model is described in section 3.5. Section 3.6 describes how the final deep learning model is evaluated for the purpose of comparing the performance to other IDS. Lastly, section 3.7 described how the validity of the thesis is maintained to ensure that knowledge derived is applicable to the area of research.

## 3.1 Scope

Research questions are attempted answered through the use case of developing a dataset which simulates the behaviour of an electrical substation. Simulation is often found in mimicking the behaviour of a network by the use of tools such as virtual machines, packet tracer. The approach proposed for this thesis presents a simulation method where the environment is defined by script given calculated features which operate differently dependent on the current scenario it is exposed to. The simulated dataset, developed for use in training a ML model, contains scenarios of both normal operation and malicious behaviour. The scenario of normal operation is defined by having an average expected value for each of the features with some noise which introduces some variation. Attack scenarios are similarly defined by having the values of the features adjusted to a varying degree. The features presented in the dataset are reflecting operational measurements of the environment. Attack scenarios presented are relevant to the environment of CI. Each scenario is created to affect the measurements in different ways and to different degrees. Though it is attempted to create the scenarios as accurately to the real environment as possible by basing the scenarios on previous work and impact analysis, the scenarios will to a greater extent be based on assumptions in how the environment might react. Measurements will be attempted to be adjusted to a representative degree, however it is not a goal of this thesis to develop a dataset which represents the behaviour of a particular substation or develop a model which can be used in one. The simulated dataset will be used to train and evaluate a Long Short-Term Memory (LSTM) classification model. The developed model will be evaluated on how capable it is in detecting the different scenarios in an attempt to understand the traits and limitations of the trained model. The contribution from this thesis is an evaluation of the described methodology of using simulation for the development of a dataset. The thesis wish to explore the benefits and weaknesses of simulation of CI. It is attempted to understand how use of a simulated dataset can help improve and iterate on ML models. The goal is to contribute in identifying methods of improving detection and prevention of malicious behaviour in the challenging sector of CI.

## 3.2 Limitations

Due to the sensitivity in data located in Critical Infrastructure, it proved to be difficult to gain access to and utilize this type of data for the purpose of the thesis. In the case that it would be used, it would have been difficult to publish the resulting model and dataset as there is a thorough process of validation to ensure that no data containing sensitive information is leaked. Maintenance of the data is necessary to prevent bad actors from gaining insight into how the environment operates, as this could lead to them potentially finding weaknesses or vulnerabilities. This means that, in using the data based on CI, utilization of collected data would be strict, limiting what it could be used for and what results could be shared. As a consequence, the results and proposed methodology would have been as difficult to verify and validate, essentially making resulting contribution of no value to future research. Measurements presented in this thesis is for this reason not derived from the actual environment of CI. The simulated dataset which is developed from this thesis, rather than reflecting accurate measurements from an operational setting, simulates a scenario where it is assumed that included features can indicate malicious behaviour. Both behaviour and reaction to attack scenarios are assumed and does not reflect real-world measurements of real electrical sub stations. The use case attempts to depict a general process in use of simulation in depicting an environment with known features. Due to limited knowledge in the field of electric power, features and measurements presented are not made to portray accurate measurements or behaviour.

The thesis originally planned to correlate usage the simulated dataset with a testbed for a greater understanding of how the two methods compare. This ultimately did not turn out to be possible and has been described as a suggestion in future work of section 6.3. Use case is therefore depicting an electrical substation using data from previous work and impact analysis. This method does however not depict the accuracy of the environment to a satisfactory degree as there were no *one* source which had data on the features under the influence of the different types of attacks. This is why the use case makes the assumption that the feature behaviour is known and capable predictors.

Attack scenarios attempted simulated are limited to attacks which affects the operation and thereby measurements of the substation environment, as only these types of attacks are relevant in using the described features in section 3.4.1. Similarly to how measurements are assumed for the operation of the environment, the environments behaviour for the duration of an attack scenario is also assumed. How the features are impacted by an attack is defined in section 3.4.2. Each of the attack scenarios are developed to showcase different behavioural changes, in an attempt to gain knowledge of what type of behaviour is not recognised by the ML model.

## 3.3 Process overview

The goal of this thesis is to create and evaluate a model of a simulated environment which could portray features extracted form CI. The first step of the process is to develop the simulated dataset and define its features. The dataset will also include different scenarios of attacks for the deep ML model to learn from. The second step is to develop and define the architecture of the deep ML model. Parameters of the model is found through iterative testing and tuning, to optimise the result of the model for it to perform to expectations. In the third step of the process the final model will be evaluated and tested. The results is set to be analysed to observe what particular scenarios were found to be hard to distinguish between and detect. Results will be compared to traditional shallow ML techniques to evaluate what benefit a deep model would have in classification. Lastly an overall analysis of the methodology will be performed to evaluate whether simulating a dataset can reflect the environment of CI. Figure 4 pictures each step of the process for this thesis.

## 3.4 Dataset development

The first step of the methodology is to develop the dataset which portrays the use case of the electric substation under the influence of cyber attacks. This is done through simulation by calculating values for each feature based on the defined scenarios in script. Though this does not create a dataset which represents the real world values of a substation, the process describes a methodology for simulating the behaviour of an environment where the assumed relevant features and their values are gathered. The purpose of the dataset is to train a ML model on it, evaluating the capabilities of the model by how well each scenario defined are classified correctly. In that the behaviour of the simulated dataset is known, limitations and weaknesses of the model can be identified, defined and adjusted.

### 3.4.1 Features

For each scenario, being normal behaviour and each type of attack, each feature is defined by an average value as well as a variation. Measurements are made by using the average and variation to generate a value where the variation defines the maximum and minimum distance from the average. Each measurement made has a random adjustment made to its average value within the distance of variation at random.

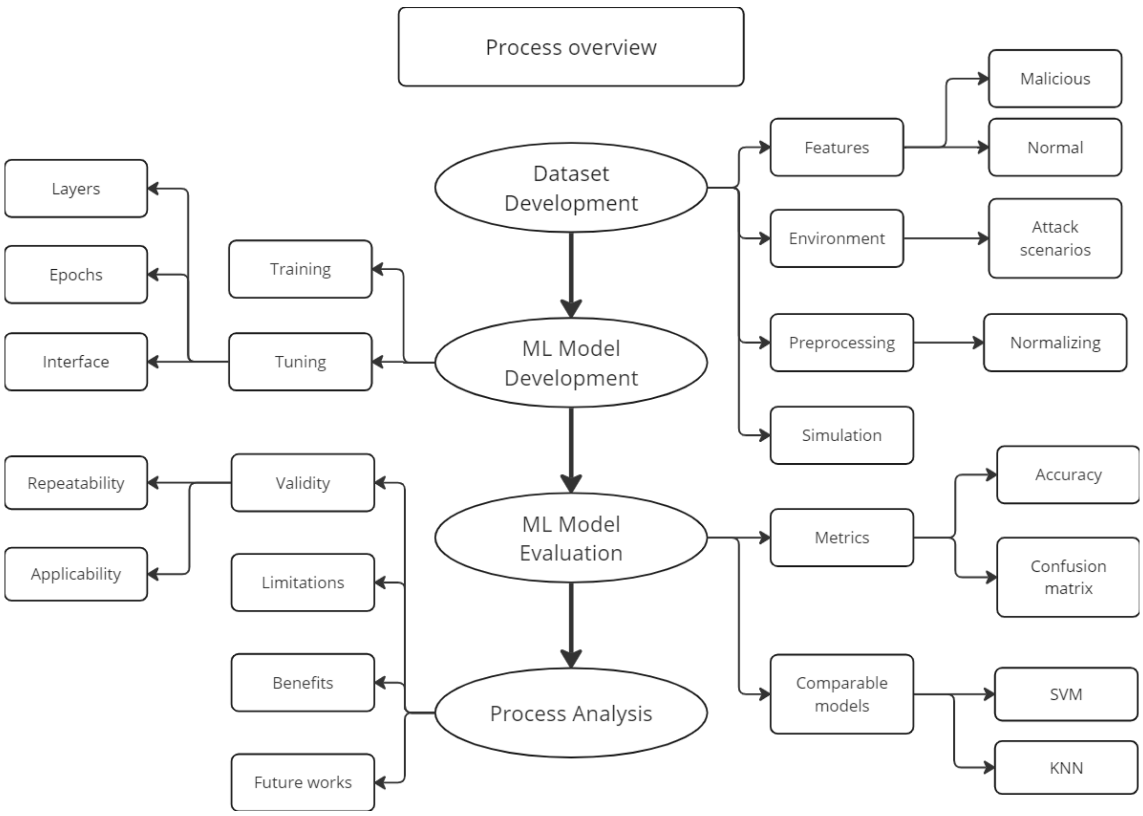


Figure 4: Modeled process overview for the thesis.

For ease in managing and deriving knowledge from each scenario in an attempt to answer the stated research questions 1.4, it is chosen to use only three features in identifying an attack. As mentioned in 2.2.7, it could be hard deriving knowledge from deep learning, due to them operating in a high dimentionality. A fewer number of features, and predefined description of how each scenario affect the features will help understand what it is the ANN sees. This has the benefit of making the final ML model more transparent in understanding how it functions and what patterns it detects. This is necessary to understand how well the proposed ML model is suited for the given scenario and the environment of CI.

The features chosen are given the name and attempted mimicking measurements found in electrical substations. Gathered from [66]–[68] it is observed that *frequency*, *current* and *voltage* are measurements which are impacted by cyber attacks. For the sake of building a scenario, these are the features which are mimicked for the simulated dataset. In a real environment these features would maintain a relationship as they impact each other. However for this scenario the features are not implemented as such and are simulated independently from each other. Table 1 defines the average and variation for each feature in normal operational conditions.

| Scenario (label) | Frequency | | Current | | Voltage | |
|---|---|---|---|---|---|---|
| | Avg | Var | Avg | Var | Avg | Var |
| Normal (0) | 50.0 | 0.02 | 400.0 | 2.0 | 150.0 | 1.0 |

Table 1: Normal behaviour of simulated features defined by the average and variation.

### 3.4.2 Attack scenarios

Each attack scenario is created to have a different distribution of the values witch is presented in table 1, to simulate the changed state of the environment. The variants are a mixture of increased and decreased average and variation value, to different degrees. Each scenario is attempted to reflect the intrusiveness of actual attacks relevant to the field of CI. However, as mentioned in section 3.2 it is assumed how particular attacks affects operational measurements. Attack scenarios are therefore also limited to attacks which are presumed to have an impact on the operation as this is when they can be measured for given the methodology of this thesis. Therefore, scenarios created mostly reflect the last stages of the *cyber kill chain*, where the actor is taking action to achieve his objective.

The dataset and attack scenarios presented in it, should be created such that it answer the question of what the capabilities of the ML model is. Scenarios presented should in varying degree be intrusive and stealthy to determine what the final model can distinguish between. What is important is that the different scenarios reflect different ways attacks could impact the features to potentially further evaluate what variations are stealthy enough to bypass the detection system. Each scenario is named by their level of impact. This level does not describe, the overall consequences that particular attack has over the environment, rather to what degree they affect the measured features.

**Scenario 1: High impact denial of service**
For this scenario, an attacker has an established foothold in the control zone of the ICS environment. The attacker use his privileges to manipulate the packet flow of orders which disrupt normal operation. The high impact is seen by larger variance in the frequency and adjustments to both the current and voltage, average and variance. Table 2 presents the values for simulating the features.

| Scenario (label) | Frequency | | Current | | Voltage | |
|---|---|---|---|---|---|---|
| | Avg | Var | Avg | Var | Avg | Var |
| High impact DoS (1) | 50.0 | 0.05 | 200.0 | 50.0 | 100.0 | 10.0 |

Table 2: Simulated behaviour under the influence of a DoS attack.

**Scenario 2: High impact process manipulation**
This scenario describes an attack where the actor, having control over the ICS environment, use packet manipulation to disrupt certain areas of the operation. The attack impacts the current to a high degree, lowering the average output and increasing the variance. Table 3 presents the values for simulating the features.

| Scenario (label) | Frequency | | Current | | Voltage | |
|---|---|---|---|---|---|---|
| | Avg | Var | Avg | Var | Avg | Var |
| High impact process manipulation (2) | 50.0 | 0.02 | 100 | 10.0 | 150.0 | 1.0 |

Table 3: Simulated behaviour under the influence of a process manipulation attack.

**Scenario 3: Medium impact process manipulation**
An attacker has infiltrated the ICS environment and is maintaining presence as a MITM. From this position the actor is able to disrupt processes related to operation. Operational measurements are impacted by a higher average frequency and voltage. Table 4 presents the values for simulating the features.

| Scenario (label) | Frequency | | Current | | Voltage | |
|---|---|---|---|---|---|---|
| | Avg | Var | Avg | Var | Avg | Var |
| Medium impact process manipulation (3) | 50.02 | 0.01 | 100 | 10.0 | 180.0 | 1.0 |

Table 4: Simulated behaviour under the influence of a process manipulation attack of medium impact.

**Scenario 4: Medium impact replay attack**

From a position within the ICS environment, an attacker is repeating previously sent packages which disrupt some physical processes. This causes instability in the operation, leading in this case an increase in current and voltage with higher variance. Table 5 presents the values for simulating the features.

| Scenario (label) | Frequency | | Current | | Voltage | |
|---|---|---|---|---|---|---|
| | Avg | Var | Avg | Var | Avg | Var |
| Medium impact replay attack (4) | 50.0 | 0.02 | 500 | 20.0 | 200.0 | 5.0 |

Table 5: Simulated behaviour under the influence of a replay attack.

**Scenario 5: Low impact stealth attack**
This scenario describes an attacker with a presence in the ICS environment. In an attempt to stay hidden the measurements made are manipulated to show operation as normal. The resulting measurements are therefore very similar to normal, other than that the variance is lower. Table 6 presents the values for simulating the features.

| Scenario (label) | Frequency | | Current | | Voltage | |
|---|---|---|---|---|---|---|
| | Avg | Var | Avg | Var | Avg | Var |
| Low impact stealth attack (5) | 50.0 | 0.01 | 400 | 1.0 | 150.0 | 0.8 |

Table 6: Simulated behaviour under the influence of a stealth attack.

**Scenario 6: Low impact direct damage attack**
The direct damage attack similarly to the stealth attack is attempted conducted in secrecy, by only impacting operation by small margins. In difference to the stealth attack the environment is attempted to be put in a state, which over time would tear and degrade its components. Operation is measured as mostly normal, other than slight increases in variation. Table 7 presents the values for simulating the features.

| Scenario (label) | Frequency | | Current | | Voltage | |
|---|---|---|---|---|---|---|
| | Avg | Var | Avg | Var | Avg | Var |
| Low impact direct damage attack (6) | 50.0 | 0.03 | 400 | 3.0 | 150.0 | 1.5 |

Table 7: Simulated behaviour under the influence of a stealthy direct damage attack.

The full dataset is built by simulating multiple iterations of the defined scenarios in randomized order to not introduce a bias in that the model recognizes the order of operation rather than classify based on the given features. The dataset is simulated in two different instances, one for use in training of the model and the other for use in testing the trained model. To simulate a consistent state of the environment for periods of time, each iteration of a scenario has a defined number of measurements before the state changes. The final model was thereby composed of the seven different scenarios where each measurement was simulated a thousands times per scenario and each scenario was iterated another thousands times for the training set. The test set was made smaller in that two hundred iterations of each scenario was simulated, where each scenario also had a thousands measurements. In training of the model, 20 percent of the training set is used as a validation set.

To further emphasize and help the model learn that any other attack could potentially come after any given state and prevent the dataset from randomly cluster similar scenarios, randomization was performed by grouping the defined scenarios before randomizing their order to be picked. When every scenario had been picked a new order of the scenarios are created and simulated.

## 3.5 Machine Learning model development

The second step of the methodology is developing the model to be used for classification of the created dataset. The trained model is evaluated by the degree of which it is capable of distinguishing the labels of the dataset. Some of the scenarios are created such that they will be difficult to differentiate, in that the values of the behaviours are overlapping. A goal of the model is therefore to be able to use previously read samples to gain context in determining the current prediction.

### 3.5.1 Model requirements

In the case of a real cyberattack, it is important that the event is alerted as soon as possible. This will help in responding to the attack and limit potential damage an adversary could do to the systems. This is especially true for CI, where an impact could have major consequences to those in need of the resources provided. Detection methodology used for intrusion detection should therefore be in real time and be able to identify an attempt of attack early. The ICS environments are prone to high degree of noise which could impact the performance of the IDS, making it harder to identify malicious behaviour. Resilience should therefore be taken into consideration in the development of such a system.

### 3.5.2 Model architecture

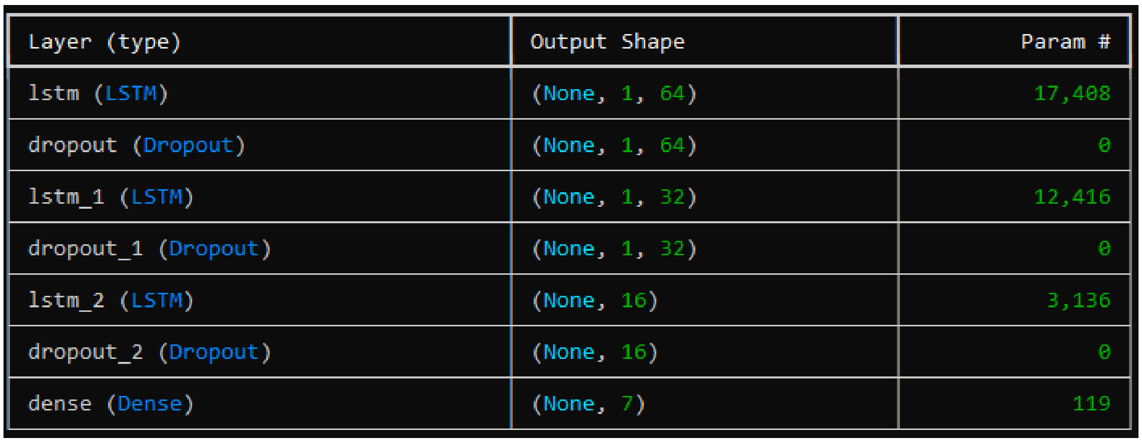
The ML model to be developed for this thesis is a Recurrent Neural Network (RNN) with Long Short-Term Memory (LSTM). This has been chosen due to its functionality in taking previous data into consideration when deciding the current state of the environment. The hypothesis is that the model will be able to detect even the stealthy attacks. Using the context of previous data it could potentially observe the trend of values being normal, yet collectively different from the normal behaviour. For developing the model, the Keras library from TensorFlow is used in python together with the sklearn library for data handling.

Parameters of the ML model will be tuned through iterations of training, initially using the accuracy of the model in deciding what values are optimal to the performance of the model. The initial model architecture is derived from the work of [16], where an LSTM model is developed for the purpose of network intrusion detection. Both a binary- and multi-classification model will be trained on the simulated dataset to see how well it performs in general detection as well as what particular activities are found to be challenging.

For the binary classification architecture, the model will make use of *Sigmoid* as the activation function, *binary crossentropy* as the Loss function and the *adam* optimization algorithm. The multiclass classification model will use *softmax* for its activation function, *categorical crossentropy* for its loss function and *adam* as the optimization function.

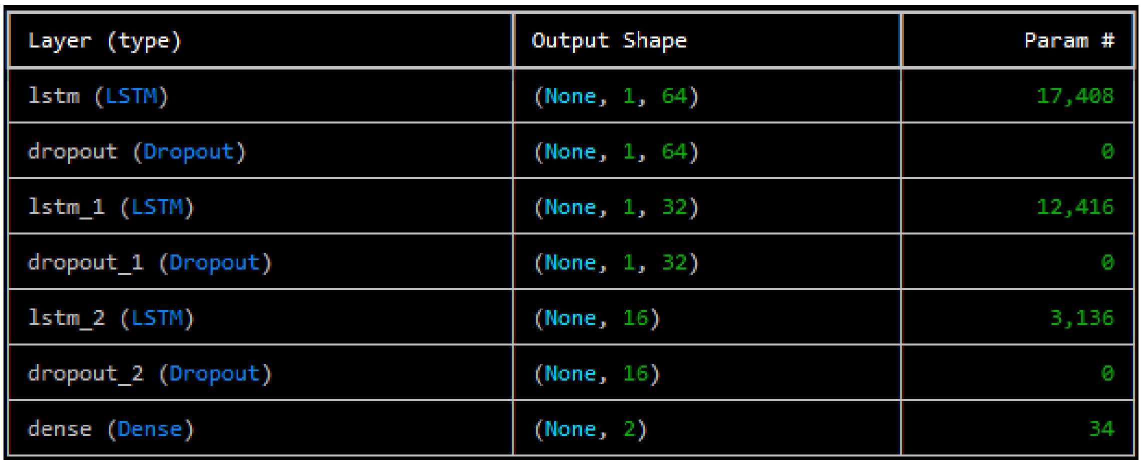
The proposed model is composed of three LSTM layers, each followed by a *dropout* layer to counteract overfitting. The last layer is densely connected and outputs the predicted class. Two modes are created, one which outputs a binary classification, where sigmoid is used as the activation. The other model is multiclass classification and predicts the particular attack using softmax as the activation function.

Figure 5 displays the architecture of the multiclass classification model and its trained parameters. Figure 6 displays the architecture and parameters of the binary classification model. The configuration of the model is further discussed in section 4.4, where the initial model is evaluated and analysed to improve its future iterations. At this stage the optimal number for batch size, dropout frequency and units per layer, is described.



```
Layer (type)              Output Shape         Param #
lstm (LSTM)               (None, 1, 64)        17,408
dropout (Dropout)         (None, 1, 64)        0
lstm_1 (LSTM)             (None, 1, 32)        12,416
dropout_1 (Dropout)       (None, 1, 32)        0
lstm_2 (LSTM)             (None, 16)           3,136
dropout_2 (Dropout)       (None, 16)           0
dense (Dense)             (None, 7)            119
```

Figure 5: LSTM multiclass classification architecture.



```
Layer (type)              Output Shape         Param #
lstm (LSTM)               (None, 1, 64)        17,408
dropout (Dropout)         (None, 1, 64)        0
lstm_1 (LSTM)             (None, 1, 32)        12,416
dropout_1 (Dropout)       (None, 1, 32)        0
lstm_2 (LSTM)             (None, 16)           3,136
dropout_2 (Dropout)       (None, 16)           0
dense (Dense)             (None, 2)            34
```

Figure 6: LSTM binary classification architecture.

### 3.5.3 Pre-processing

Pre-processing is the process of preparing the collected raw data for usage in training the model. Data which is collected from the environment has to be adjusted and cleaned so that it can be inserted into the model as well as give beneficial input to the training.

Since each of the features operate on a different magnitude of scale, larger numbers introduced to the model during training might have larger impact on the set weight than the other values. The features should therefore be normalized to ensure that all features are considered equally during training. Values are normalized by rescaling the values to fit between 0 and 1.

The labels of the dataset is defined by a number ranging from 0 to 6 for multiclass classification and 0 to 1 in binary classification. For the multiclass classification to interpret the label correctly is should be converted from an integer to a categorical label by the standards which is required by Keras. Using LabelEncoder each label is converted to a binary class matrix.

For most trained classification models, the dataset which is presented during training is shuffled to a random sequence before it is used as to not introduce bias in how it is ordered. However, the goal of the model which is developed for this thesis is to recognize the current class, by what previous behaviour looked like. Therefore the order of sequence which values occur is relevant to the classification. The dataset is therefore not shuffled before training, but is rather randomised by their order of scenarios as described in section 3.4.2.

## 3.6 Evaluation

The evaluation of the ML model will dictate how well an IDS built on it would detect suspicious behaviour. The metrics are essential in deciding if the model is of benefit to security monitoring. A model that alerts too often would issue too many alerts, generating too much noise to find the real threats. A model issuing too few alerts, risks not detecting actual attempts of intrusion. Evaluation of the ML model is performed by statistical measurements of predictions made by the model on labeled data it has not yet been exposed to. The results of the evaluation will ultimately help dictate whether the dataset features or model architecture are capable or optimal for classifying malicious behaviour. Evaluation makes comparison between different IDS possible. It can thereby be determining what models, methodologies, datasets and features are optimal to use in different circumstances and environments.

In an attempt of understanding how the resulting ML model functions, its temporal decision making will be analysed as part of the evaluation. Given that LSTM is used, the order of events could impact how the model behave. It can be beneficial to see if there are particular arrangements of scenarios proven to be beneficial or disruptive to the performance. This will be analysed by graphing and comparing the predicted and true labels, attempting to observe any patterns. Graphs are created so that they portray different scales such that; one particular scenario can be analysed, a segment of the test-set portraying multiple subsequent scenarios can be analysed and a graph portraying a larger overview over the test-set.

### 3.6.1 Measurements

The final iteration of the developed decision making model for identifying malicious activity, will be evaluated using state of the art statistical measurements. The ML models are evaluated on a dataset which it has not previously seen in the training data, but consists of the same defined scenarios of normal behaviour and attacks. As is mentioned in [1] methodologies and measurements used for evaluation varies widely between the different research projects. This has made comparison between the IDS's difficult as there are different collections of metrics used. There is no defined standard for what metrics should be used, however some metrics are more wide spread. These metrics are chosen as they are frequently used in other research evaluating the performance of IDS, making comparisons to the developed model possible. Predictions made by the ML model is measured by the following metrics.

| | Predicted as normal activity | Predicted as malicious activity |
|---|---|---|
| Actual normal activity | True Negative (TN) | False Positive (FP) |
| Actual malicious activity | False Negative (FN) | True Positive (TP) |

Table 8: Confusion matrix.

**Confusion matrix**
Firstly the predicted classifications determined by the model is logged to a *confusion matrix*. The confusion matrix holds information on how the predictions are distributed amongst the four resulting terms [4]:

1. **True Positive (TP):** The prediction made is that the features inherits malicious activity and the features are labeled as such.
2. **True Negative (TN):** The prediction made is that the features does not entail malicious activity, and the features are labeled as normal activity.
3. **False Positive (FP):** The prediction made is that the features inherits malicious activity, however the data is labeled as normal activity.
4. **False Negative (FN):** The prediction made is that the features does not entail malicious activity, however the data is labeled as malicious.

**Accuracy**
The overall accuracy of the decision making model is found by adding up the correctly predicted instances (being TN and TP), and dividing it on the total number of samples.

$$Accuracy = \frac{TP + TN}{nr.of samples} \tag{1}$$

**Precision**
The precision gives an indication how often, when a sample is predicted as malicious activity, it is actually malicious. This can be especially beneficial in intrusion detection to determine the rate of false positives. A low level of precision could waste a lot of resources on misleading alerts.

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

**Recall**
The recall, also called the True Positive Rate (TPR), looks at the ratio in which the model is able to predict malicious behaviour when being attacked. Thereby, dividing the correctly predicted attacks (TP) on all attacks.

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

**F1 score**
The F1 score, also known as f-measure, combines the recall and precision to one formula accounting for unbalanced samples.

$$F1score = 2 \cdot \frac{recall \cdot precision}{recall + precision} \tag{4}$$

## 3.7 Validity

The validation of the project determine how well the results of the thesis reflect, and can be applied to the real world. Limitations of the thesis means that aspects of the project does not entirely portray the targeted environment or goal. This is taken into consideration when determining the results of the final product and conclusion of the thesis. Decisions and alternatives that has been made to account for the limitations are discussed and reflected upon. The methodology of the thesis is argued in how well it reflects its targeted environment and how well a result of the thesis applies to real world environment.

Though the thesis attempt to reflect how CI is impacted by the use of a simulated dataset, a limitation of the thesis is that an environment which portrays ICS was not available for testing. The thesis therefore assumes a definable and known environment for the purpose of simulating that environment. The environment itself and the attack behaviours simulated does not attempt to portray the real-world behaviour, but use the use case to explore how this could enhance the development and evaluation of a ML model for the defined environment. Validity of the thesis is maintained by first gaining a familiarity of the area of machine learning and critical infrastructure. For the development of the model, evaluation of the model is performed by the use of *state of the art* measurements which can be used for comparison to previously developed models, as to build on previous knowledge rather than starting from scratch. The lack of experience in developing a deep learning model is also recognized, which is why it was chosen to base the model on a previously developed LSTM model. Evaluation of the model is also somewhat restricted in regards to how results can be compared to previous work due to the lack of previously developed models built on datasets portraying CI, and due to the methodology of how the dataset was developed for this thesis. The results are therefore not compared to any particular works of previously developed IDS models, but the thesis considers the level of performance which should be expected by the standards of previously developed models. Literature review is also used to gain insight to CI, to best relate the context of simulation and deep learning to the area. Limitations of the thesis are clearly stated, which help declare what boundaries the thesis is confined to. Areas possessed of limitations and therefore uncertainties are for that reason described as future work. It should be recognized that this thesis is just a small step in the continuous process of research concerning IDS. Validity of the thesis is maintained by clearly stating these boundaries and limitations which this thesis has been impacted by, leaving uncertainties to future work. For future validation and enabling repeatability of the experiments performed in the thesis, the code created for the purpose of this thesis is included in the appendix.

# 4 Experimental results

This chapter presents the results of the performed tasks which were described in section 3. Firstly section 4.1, present the simulated dataset. Section 4.2 presents the results of the trained binary and multiclass Long Short-Term Memory (LSTM) classification models. Section 4.3 presents the results of evaluation performed on the comparing ML techniques. Finally, the results of the evaluation will in section 4.4 be analysed and used to understand the current models weaknesses and attempt to use this for the purpose of improving it. The results of the improved models will also be presented.

## 4.1 Resulting simulated dataset

The defined scenarios from section 3.4.2 was implemented into a script which resulted in two datasets. The dataset for binary classification has an even distribution of malicious behaviour, defined in table 2, 3, 4, 5, 6, 7, and normal behaviour as defined in table 1. The multiclass classification dataset had an even distribution over all the defined scenarios. Each dataset is also split into a training set and a test set. The dataset for training the binary and multiclass model as 7 008 000 and 12 013 000 samples respectively. The reason for the datasets being as large as they are is to accommodate for the models which are able to use the context of previous samples or scenarios in predicting the current sample. Therefore every order possible of scenarios was wanted in the dataset as to not introduce bias in that the dataset indicating that certain scenarios only appeared after another. Though this include redundant samples it is also something which can be adjusted for in the pre-processing of the dataset for future models. The test sets of the binary and multiclass datasets are composed of 2 403 400 and 1 402 400 samples respectively.

## 4.2 LSTM model performance

The final trained ML models are evaluated using a separate subset of the dataset to evaluate the model on data which has not been used during training. The test set does include the same characteristics as the training set, but is separate to evaluate whether the model has found distinguishable features or simply memorised the occurrence of labels from the training set.

Following are the resulting metrics of evaluation for the multiclass and binary classification models.

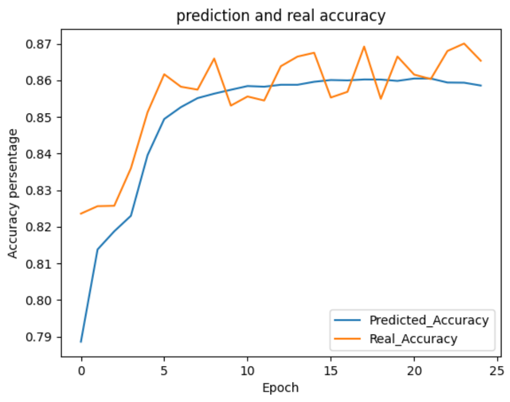
### 4.2.1 Multiclass classification

Figure 7a displays the improvements of the predicted accuracy and real accuracy for the duration of its training. Figure 7b, displays how the predicted and real loss value decreases for each epoch, indicating the progression in learning for each step.

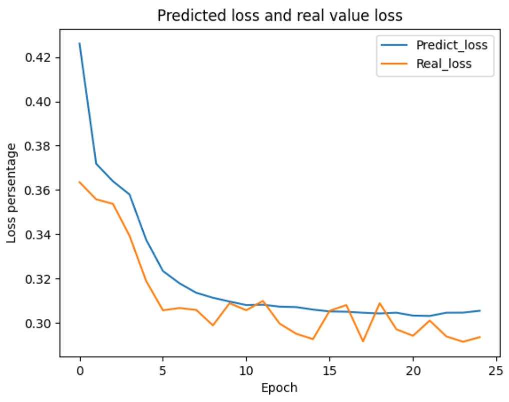
The trained multiclass classification model performed well in identifying the scenarios of attack 1-4. Following the values provided in the confusion matrix in figure 8, normal behaviour was confused with attack scenario 5. Attack scenario 6 was similarly confused with both normal behaviour and behaviour of the attack scenario 5. Regarding attack scenario 5, though most instances of the attack scenario was identified, it resulted in a lot of false positives. This indicates a poorly placed threshold, separating normal behaviour from attack scenario 5.

Though each label is represented equally in the mode, normal behaviour is put up against two scenarios which highly mimic its own behaviour. This leads the model to think that there is a higher likelihood the behaviour of being a type of attack, rather than of benign origin. Distribution of labels could have been improved introducing more values of the normal behaviour.

Table 9 contains the results of the 'classification report', displaying the precision, recall and f1-score for each class, as well as the overall accuracy, macro and weighted average. The high false positive rate of attack scenario 5 is indicated in the low precision value. Similarly, attack scenario 6 has a lower recall value due to the higher false negative. Normal behaviour was the overall hardest



(a) Predicted and real accuracy over epochs in training of the LSTM multiclass classification model.



(b) Predicted and real loss over epochs in training of the LSTM multiclass classification model.

Figure 7: Predicted and real of loss and accuracy during training the LSTM multiclass classification model.

| | Precision | Recall | f1-score | Support |
|---|---|---|---|---|
| Class 0 | 0.65 | 0.53 | 0.58 | 201 401 |
| Class 1 | 1.00 | 1.00 | 1.00 | 200 000 |
| Class 2 | 1.00 | 1.00 | 1.00 | 200 000 |
| Class 3 | 1.00 | 1.00 | 1.00 | 200 000 |
| Class 4 | 1.00 | 1.00 | 1.00 | 200 000 |
| Class 5 | 0.61 | 0.96 | 0.75 | 200 000 |
| Class 6 | 0.92 | 0.57 | 0.70 | 201 000 |
| Accuracy | | | 0.86 | 1 402 401 |
| Macro avg | 0.88 | 0.87 | 0.86 | 1 402 401 |
| Weighted avg | 0.88 | 0.86 | 0.86 | 1 402 401 |

Table 9: Classification report of LSTM multiclass classification model predictions.

class to identify, indicated by the precision, recall and f1-score. The overall accuracy of the LSTM multiclass classification model turned out to be 86%.

In attempting to observe a pattern of behaviour in how the multiclass classification model operate, the graphs portrayed in figure 9 pictures the predicted and true label made from the first samples of the test set. It was however not found any correlation or patterns of occurrences in falsely classified samples as there would have been if the model took the previously processed samples into its consideration.

### 4.2.2 Binary classification

Figure 10a and figure 10b presents the progression in accuracy and loss for each epoch in training of the binary LSTM classification model. Compared to the multiclass classification model, the plot reaches a higher accuracy much quicker, before the plots plateaus before reaching the performance of the multiclass model. Compared to the multiclass model, the confusion matrix of the binary model displayed in figure 11, show there were no false positives. The binary model does not entail the same problem of having a grate portion of false positives in labeling normal behaviour. This indicates a distribution of labels which benefit the prediction of normal behaviour.

However some of the malicious activity was classified as normal. Looking into the false negatives, the classes which were wrongly classified was mostly the attack scenarios 5 and 6, as seen in table 10. The binary model was not able to identify any of the attacks from scenario 5, and missed most attacks from scenario 6.

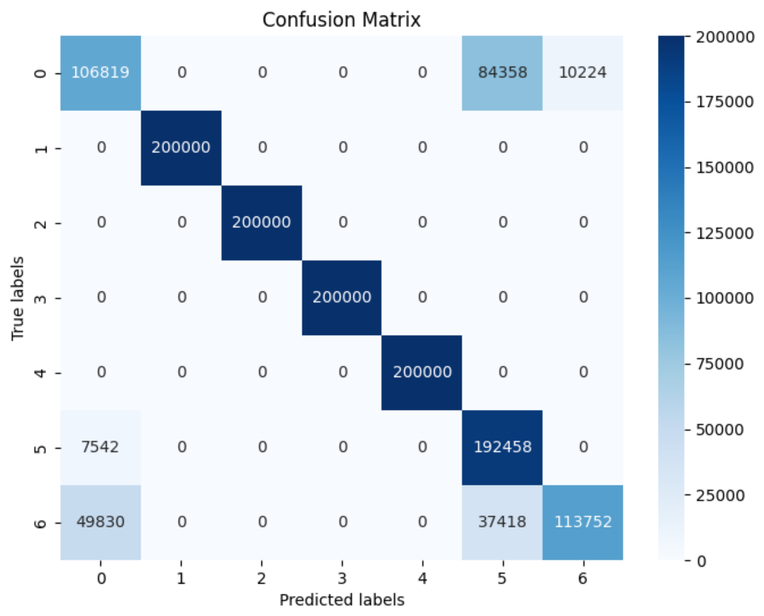


Figure 8: Confusion matrix LSTM multiclass classification.

| Label | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|
| Falsely classified | 0 | 0 | 0 | 0 | 0 | 200 000 | 141213 |

Table 10: Wrongly classified labels of the binary LSTM classification model.

Table 11 shows the resulting classification report from the models predictions of the test dataset. A lower precision in the benign class and a lower recall of the malicious class is the result of the false negatives.

## 4.3 Comparable ML approaches

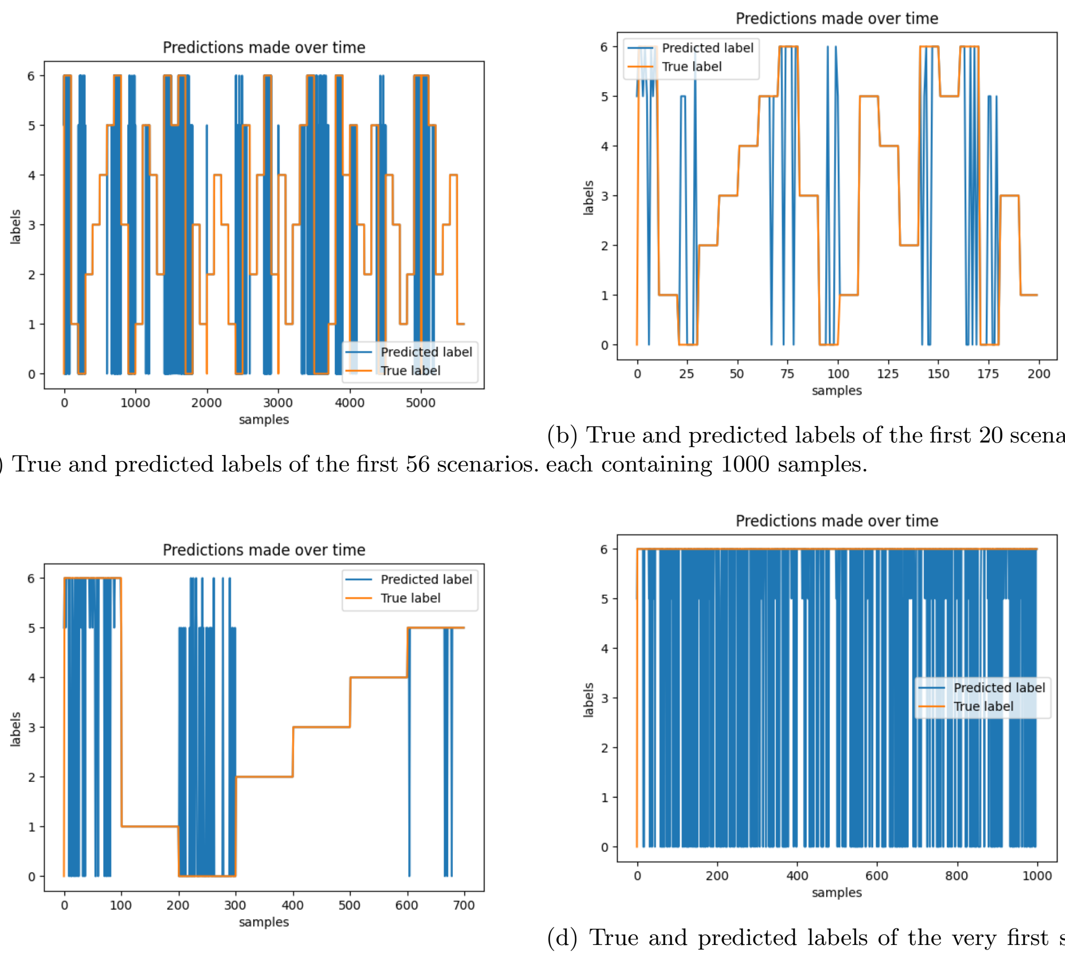
As a point of reference, two other ML approaches were chosen to give a comparison to the performance of the LSTM model. The KNN and SVM algorithm was trained and tested on the same dataset. Due to conflicts in handling the size of the original dataset using sklearn's library for SVM and KNN, the models was trained on the 90 000 first samples of the training set, and tested on the 28 000 first samples of the test set.

### 4.3.1 k-Nearest Neighbors

Figure 12 displays the confusion matrix over the predictions made by the KNN multiclass classification model. The model, similarly to the LSTM found well defined thresholds for attack scenarios

| | Precision | Recall | f1-score | Support |
|---|---|---|---|---|
| Benign | 0.78 | 1.00 | 0.88 | 1 203 401 |
| Malicious | 1.00 | 0.72 | 0.83 | 1 200 000 |
| Accuracy | | | 0.86 | 1 402 401 |
| Macro avg | 0.89 | 0.86 | 0.86 | 1 402 401 |
| Weighted avg | 0.89 | 0.86 | 0.86 | 1 402 401 |

Table 11: Classification report of binary LSTM model predictions.



(a) True and predicted labels of the first 56 scenarios.

(b) True and predicted labels of the first 20 scenarios, each containing 1000 samples.

(c) True and predicted labels of the first 7 scenarios.

(d) True and predicted labels of the very first scenario, being of attack scenario 6.

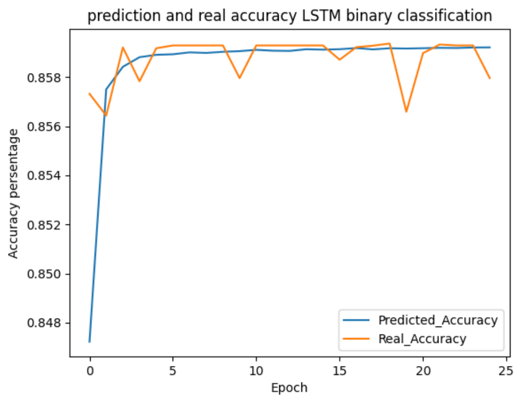
Figure 9: Timelines over predictions made in testing of the multiclass classification model. Graphs are shown from different perspectives number of samples. Inconsistency in displayed number of samples is due to reducing the number of output samples for clarity of graph.

one through four. Also, similarly to the LSTM, normal behaviour contra attack scenario five and six, was difficult to distinguish. This was to be expected due to the latter attack scenarios mostly operating in the same range as benign behaviour.

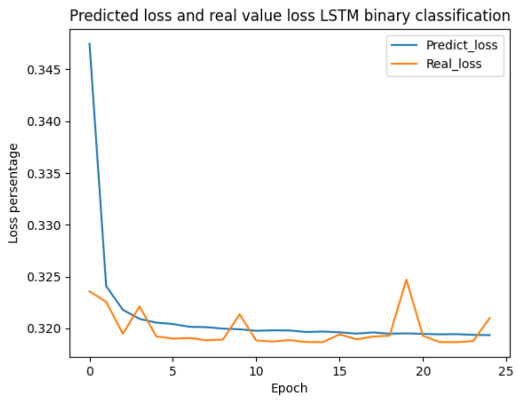
The performance of the KNN algorithm is shown by its accuracy, precision, recall and f1-score in table 12. Looking at both normal behaviour and attack scenario 5 and 6, the algorithm does not reach the performance of the LSTM multiclass classification model. The KNN algorithm is also impacted by scenario 5 and 6 operating in the same range as normal behaviour and as the two scenarios are more heavily represented in the area, the performance in identifying normal behaviour is suboptimal.

### 4.3.2 Support Vector Machine

Figure 13 displays the resulting confusion matrix created from the predictions made by the SVM model on the test set. The SVM, similarly to the KNN and LSTM model has difficulty in distinguishing normal behaviour from attack scenario 5 and 6. Compared to KNN, the SVM algorithm is even more influenced by the class distribution. The trained hyperplane is as expected, not able to find an optimal orientation which separates the labels as they operate within the same area. The overall performance as shown in table 13, shows that the SVM algorithm performs worse than both the KNN and LSTM model for this particular environment.



(a) Predicted and real accuracy over epochs in training



(b) Predicted and real loss over epochs in training

Figure 10: Development in accuracy and loss during training of binary LSTM classification model

| | Precision | Recall | f1-score | Support |
|---|---|---|---|---|
| Class 0 | 0.53 | 0.55 | 0.54 | 4028 |
| Class 1 | 1.00 | 1.00 | 1.00 | 4000 |
| Class 2 | 1.00 | 1.00 | 1.00 | 4000 |
| Class 3 | 1.00 | 1.00 | 1.00 | 4000 |
| Class 4 | 1.00 | 1.00 | 1.00 | 3972 |
| Class 5 | 0.62 | 0.76 | 0.68 | 4000 |
| Class 6 | 0.81 | 0.59 | 0.68 | 4000 |
| Accuracy | | | 0.84 | 28 000 |
| Macro avg | 0.85 | 0.84 | 0.84 | 28 000 |
| Weighted avg | 0.85 | 0.84 | 0.84 | 28 000 |

Table 12: Classification report of KNN multiclass classification

## 4.4 Improved stateful LSTM classification model

It was from the results of the initial LSTM classification model determined that the model did not possess the trait of being able to take the context into consideration for when determining the class of a current sample. From this it was recognized that the model would probably benefit from being stateful, rather than stateless. This was used as a base to make adjustments to the model in an attempt to enhance its performance. An additional input layer was included in the model to accommodate for static batch sizes which is needed.

Multiple iterations of the models were trained and tested to find the optimal topology for the model. Statefulness was implemented and tested on a combination of every layer, where it was found that it would be the most efficient on the last layer. Number of units per layer, the batch size and the dropout frequency was also adjusted iteratively to find the optimal configuration. Table 14 depicts different configurations of the statful LSTM classification models and how they performed in comparison to each other. Tests were performed on the multiclass classification model, before also being applied to the binary classification model. The configuration that was ultimately chosen was a batch-size of 128, with the first LSTM layer containing 64 units, being stateless and having a dropout of 0.02. The second LSTM layer contains 32 units, is also stateless and has a dropout of 0.02. The last LSTM layer has 16 units, is statefull and has a dropout of 0.02. The number of samples are shown as different in the results of the stateful LSTM model due to that the batch-size must be a factor of the number of samples.

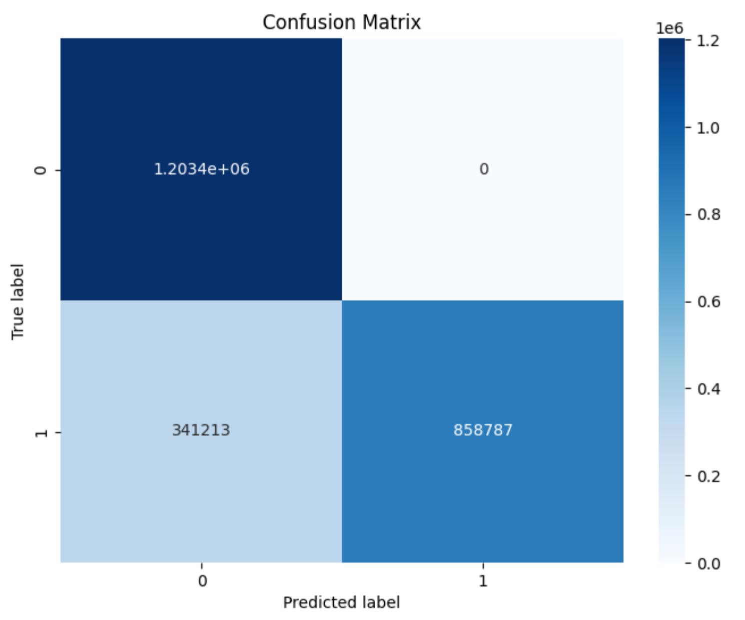
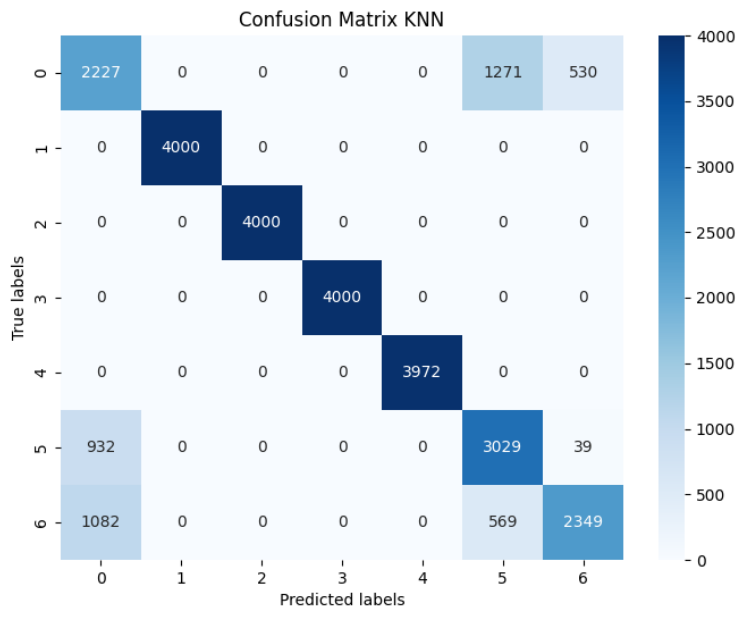


Figure 11: Confusion matrix LSTM binary classification



| True \ Predicted | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|
| 0 | 2227 | 0 | 0 | 0 | 0 | 1271 | 530 |
| 1 | 0 | 4000 | 0 | 0 | 0 | 0 | 0 |
| 2 | 0 | 0 | 4000 | 0 | 0 | 0 | 0 |
| 3 | 0 | 0 | 0 | 4000 | 0 | 0 | 0 |
| 4 | 0 | 0 | 0 | 0 | 3972 | 0 | 0 |
| 5 | 932 | 0 | 0 | 0 | 0 | 3029 | 39 |
| 6 | 1082 | 0 | 0 | 0 | 0 | 569 | 2349 |

Figure 12: Confusion matrix KNN multiclass classification

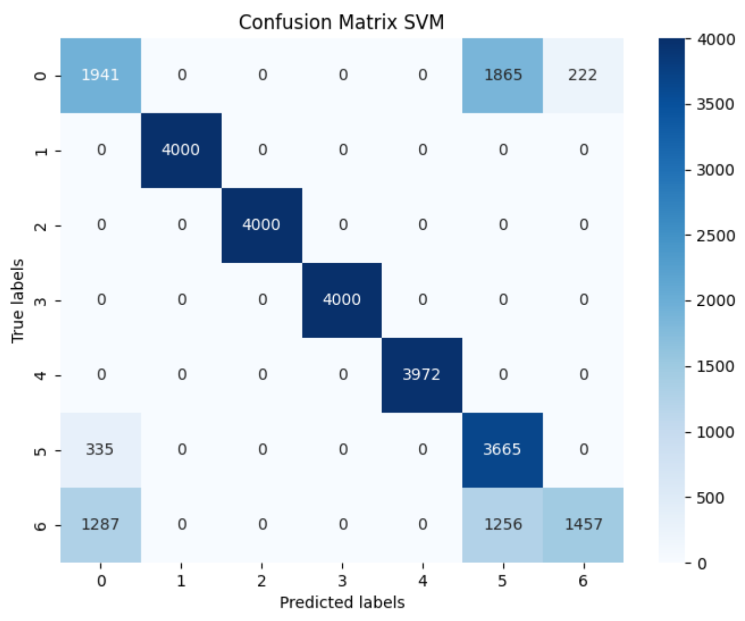


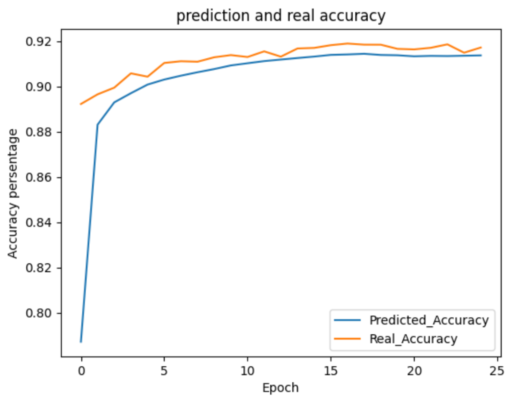
Figure 13: Confusion matrix SVM multiclass classification

| | Precision | Recall | f1-score | Support |
|---|---|---|---|---|
| Class 0 | 0.54 | 0.48 | 0.51 | 4028 |
| Class 1 | 1.00 | 1.00 | 1.00 | 4000 |
| Class 2 | 1.00 | 1.00 | 1.00 | 4000 |
| Class 3 | 1.00 | 1.00 | 1.00 | 4000 |
| Class 4 | 1.00 | 1.00 | 1.00 | 3972 |
| Class 5 | 0.54 | 0.92 | 0.68 | 4000 |
| Class 6 | 0.87 | 0.36 | 0.51 | 4000 |
| Accuracy | | | 0.82 | 28 000 |
| Macro avg | 0.85 | 0.82 | 0.81 | 28 000 |
| Weighted avg | 0.85 | 0.82 | 0.81 | 28 000 |

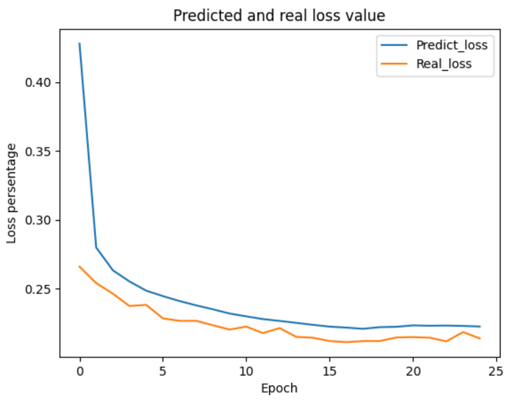
Table 13: Classification report of SVM multiclass classification

| Batches | LSTM_0 | | | LSTM_1 | | | LSTM_2 | | | Accuracy |
|---|---|---|---|---|---|---|---|---|---|---|
| | Units | Stateful | Dropout | Units | Stateful | Dropout | Units | Stateful | Dropout | |
| 256 | 32 | False | 0.01 | 16 | True | 0.01 | 8 | True | 0.01 | 0.88 |
| 128 | 64 | False | 0.01 | 32 | False | 0 | 16 | True | 0 | 0.92 |
| 64 | 128 | False | 0.01 | 64 | False | 0 | 32 | True | 0 | 0.67 |
| 128 | 32 | False | 0.1 | 64 | False | 0 | 32 | True | 0 | 0.88 |
| 128 | 32 | False | 0.1 | 64 | False | 0.01 | 64 | True | 0 | 0.91 |
| 128 | 64 | True | 0.01 | 32 | True | 0 | 16 | True | 0 | 0.72 |
| 128 | 64 | False | 0.01 | 128 | False | 0 | 128 | True | 0.01 | 0.91 |
| 128 | 64 | False | 0.01 | 64 | False | 0 | 16 | True | 0 | 0.92 |
| 128 | 64 | False | 0.02 | 32 | False | 0.02 | 16 | True | 0.02 | 0.92 |

Table 14: Different configurations of the proposed stateful LSTM classification model and the performed accuracy.



(a) Predicted and real accuracy over epochs in training of the stateful LSTM multiclass classification model



(b) Predicted and real loss over epochs in training of the stateful LSTM multiclass classification model

Figure 14: Predicted and real, loss and accuracy during training of the stateful LSTM multiclass classification model

| | Precision | Recall | f1-score | Support |
|---|---|---|---|---|
| Class 0 | 0.75 | 0.79 | 0.77 | 201 401 |
| Class 1 | 1.00 | 1.00 | 1.00 | 200 000 |
| Class 2 | 1.00 | 1.00 | 1.00 | 200 000 |
| Class 3 | 1.00 | 1.00 | 1.00 | 199 967 |
| Class 4 | 1.00 | 1.00 | 1.00 | 200 000 |
| Class 5 | 0.80 | 0.94 | 0.86 | 200 000 |
| Class 6 | 0.92 | 0.70 | 0.79 | 201 000 |
| Accuracy | | | 0.92 | 1 402 368 |
| Macro avg | 0.92 | 0.92 | 0.92 | 1 402 368 |
| Weighted avg | 0.92 | 0.92 | 0.92 | 1 402 368 |

Table 15: Classification report of stateful LSTM multiclass classification.

### 4.4.1 Stateful LSTM multiclass classification model

Figure 15, portrays the confusion matrix over the improved stateful LSTM multiclass classification model. The stateful classification model, similarly to the stateless model, also posses some of the same issues. There is a significant portion of the predicted samples which results in both false negative and false positive, though not to the same extent as the stateless model. Table 15 states the precision, recall and f1-score for each of the different classes, as well as the overall accuracy. The lowest f1-score is of the class portraying normal behaviour, indicating that the model still prioritize predicting either class five or six in cases where the overlapping feature values causes uncertainty. The resulting accuracy of the model is 92%, compared to the stateless model which was 86%. In observing the wrongfully predicted classes it is seen that, the further a scenario is played out, the more accurately the model is capable of predicting the correct class. Figure **??**, depicts a timeline over the first predicted and actual classes. From these graphs the frequency of wrongly classified samples can be observed. As seen in figure 16a, the frequency of wrongfully classified classes is reduced over the period of scenario six and zero. The first half of the first scenario, being scenario six, has a total of 213 wrong classifications. The second half of the same scenario have a total of 81 wrong predictions, which indicates that the model has taken the previously processed samples to enhance its context awareness for its future predictions.

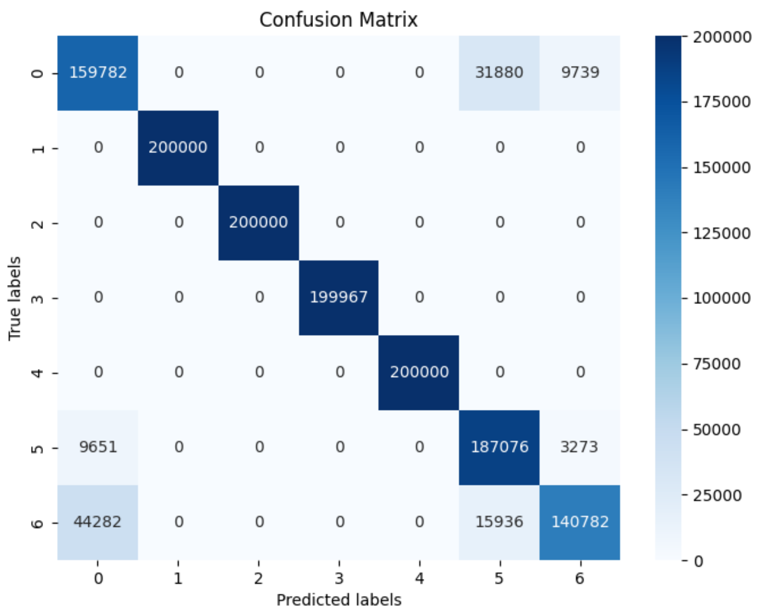


Figure 15: Confusion matrix stateful LSTM multiclass classification

| Label | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|
| Falsely classified | 77 923 | 0 | 0 | 0 | 0 | 72 771 | 61 927 |

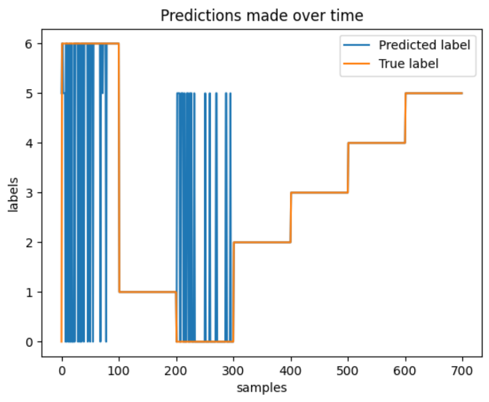
Table 16: Wrongly classified labels of the stateful binary LSTM classification model.

### 4.4.2 Stateful LSTM binary classification model

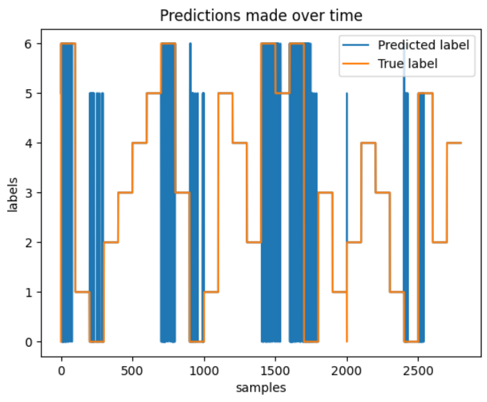
Looking at the results from testing the stateful LSTM binary classification model it is observed that it also benefited significantly from implementing statefulness. The model was capable of achieving a 91% accuracy when making predictions towards the testset as seen in table 17. The confusion matrix shown in figure 18, display the distribution of falsely and correctly classified samples. In different to the stateless model, this model does have more false negatives. However it is now capable of identifying the behaviour of attack scenario five, which it found none of in the stateless model, and scenario six to a greater extent. The weakness of the model still comes from its capability in distinguishing class zero, five and six, which can be seen in table 16 which displays what classes the wrongfully classified samples come from.

| | Precision | Recall | f1-score | Support |
|---|---|---|---|---|
| Benign | 0.89 | 0.94 | 0.91 | 1 203 401 |
| Malicious | 1.00 | 0.72 | 0.83 | 1 199 927 |
| Accuracy | | | 0.91 | 2 403 328 |
| Macro avg | 0.91 | 0.91 | 0.91 | 2 403 328 |
| Weighted avg | 0.91 | 0.91 | 0.91 | 2 403 328 |

Table 17: Classification report of binary LSTM model predictions.

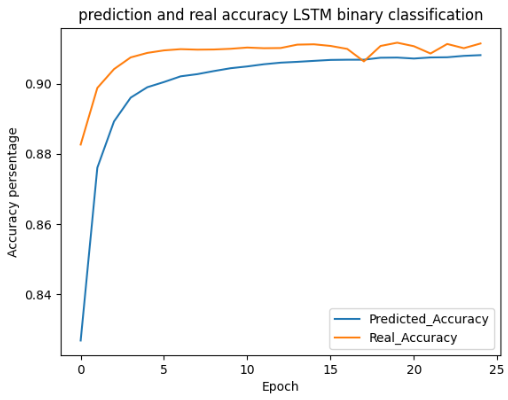


(a) Timeline over predictions made in testing of the stateful LSTM multiclass classification model of the first seven scenarios.

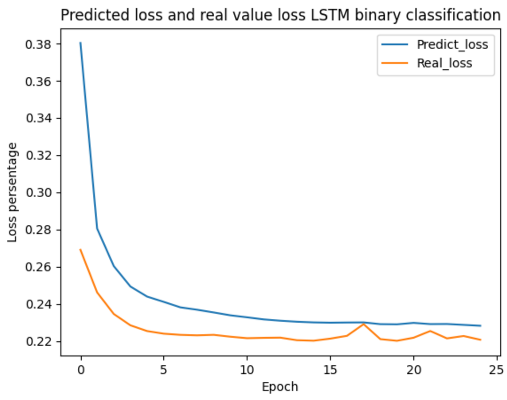


(b) Timeline over predictions made in testing of the stateful LSTM multiclass classification model of the first 24 scenarios

Figure 16: Timelines over predictions made of the stateful LSTM multiclass classification model.



(a) Predicted and real accuracy over epochs in training of the stateful LSTM binary classification model



(b) Predicted and real loss over epochs in training of the stateful LSTM binary classification model

Figure 17: Predicted and real, loss and accuracy during training of the stateful LSTM binary classification model

# 5 Discussion

For the discussion, the performance of the developed LSTM model will be discussed in regards to strengths and weaknesses and how such a model theoretically would or wouldn't function in an operational environment of critical infrastructure. Afterwards the methodology of using simulated dataset in development of a ML model will be discussed in regards to what challenges, weaknesses and strengths such an approach has on the environment of critical infrastructure.

## 5.1 Capabilities of the trained Machine Learning model

### 5.1.1 Overall performance of classification models

Results provided in chapter 4 were quite accurate to the expectations of the models. The features of attack scenario one through four was clearly distinguished from both normal behaviour and the behaviour of each other. Scenario five and six, was in contrast to the aforementioned scenarios much harder to distinguish. This was expected as they by design would operate stealthily and be

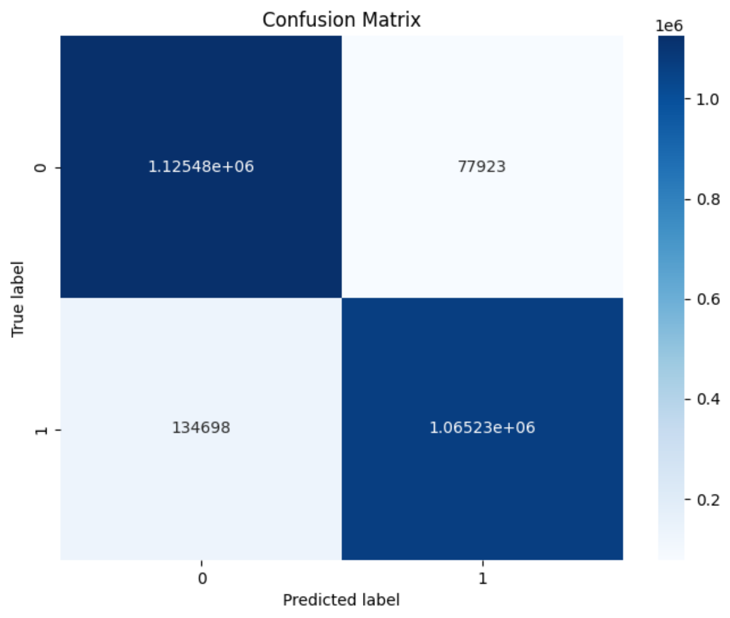


Figure 18: Confusion matrix stateful LSTM binary classification.

harder to classify. All of the classification models had difficulty in separating label zero, five and six due to the values of the features overlapping.

The initial stateless LSTM multiclass classification model presented in section 4.2.1, though able to identify most instances of attack scenario five, it came at the cost of a high amount of false positives. It is likely that attack scenario five was more frequently predicted than normal behaviour as the behaviour of the attack scenario operates within the majority of the area of normal behaviour. As attack scenario states in table 6, the variance of frequency and current is lowered to half of normal behaviour, while the variance of voltage is lowered by 0.2. Attack scenario five therefore overlaps with most of the area of normal behaviour, leading to the model mostly predicting that scenario. Introducing the training data to a greater share of normal behaviour could have improved the false positive rate, while most likely lowering the accuracy in predicting scenario five. The benefit of introducing a greater share of normal traffic is that it simulates a real environment more accurately. Attack scenario six was similarly to attack scenario five, overlapped by both attack scenario five and normal behaviour. This is thought to be the reason for the high amount of false negatives.

The stateless binary classification model presented in section 4.2.2, though not as impacted by false negatives, resulted in an overall lower accuracy compared to the stateless LSTM multiclass classification model. The dataset used for training was in comparison to the multiclass model, built on a dataset with equally many samples of benign and malicious labels rather than equally sized attack scenarios. Looking into the labels of the falsely labeled predictions stated in table 10, most of the attacks of scenario six was not detected, while none attacks of scenario five was detected. This shows the contrary point to the problem which the same scenarios had in the multiclass model. In comparison to scenario five and six, normal behaviour was over represented, further overshadowing the scenarios, leading to the model rather predict it as normal behaviour.

The traditional ML techniques, SVM and KNN, performed overall worse than the LSTM model. This was as expected, as it was known that these methodologies would have difficulty in distinguishing between scenario zero, five and six.

The improved stateful LSTM multiclass classification model, presented in section 4.4, performed much better than the initial model, mostly due to the introduction of statefulness. This gave the model context of previously processed samples in determining the current class. When determining the class of the overlapping behaviour of attack scenario five, six and normal behaviour, the model now has the capability of taking the variability of the measured values into consideration. From the results of the stateful model compared to the stateless, it is observed that the model is capable of using the given context to better classify the scenarios. The stateless model compared to the traditional ML techniques performed quite similarly, indicating that the stateless model does not in fact have the trait of context which the stateful model has.

Likewise to its multiclass counterpart, the stateful LSTM binary classification model also performed better than the stateless model. In different to the stateless model, it was capable of detecting scenarios of attack scenario five, and classified samples with an overall higher accuracy. The new iteration of the model did however introduce a higher number of false positives, most likely due to the new models capability of now considering attack scenario five as a likely class. The false positives, most likely derives from previously processed attack scenarios of class five and six, which lead the model to believe that the next benign scenario was part of the attack. Both the stateful LSTM models, after being improved upon from the initial analysis performed to much more satisfactory degree.

### 5.1.2 Capabilities of the LSTM architecture

The hypothesis of using the LSTM architecture for the DNN model, was that it would be able to distinguish between normal behaviour and attack scenario five and six. This was thought as previously read features are passed onwards as input to the next prediction. The hope was that the model would be able to distinguish the labels based on the changes of variety in the previous measurements. It was therefore never expected that every scenario would be identified correctly, but that it would be identified with more precision as more samples of the attack was measured. Looking at the performance of the stateless model, the number of attacks predicted wrong for the duration of a scenario did not decrease over the period. Further looking at the behavioural patterns of the multiclass model displayed in figure 9, there does not seem to be any clear patterns or dependencies in the decision making. The predictions made are mostly influenced by the values of current features, in contrast to how those features relate to the previous measurements.

When pivoting the model over to being stateful, it was observed that the number of wrongly predicted classes was higher in the first half of a scenario than in the second half. This indicates that as the model becomes familiar with the context of the scenario, it is much more capable of identifying those particular attacks. This trait of the LSTM model could be especially useful in an ICS environment to detect stealthy attacks. As the attacks attempt to stay hidden by making as little noise as possible, the stateful LSTM could detect the attacks from small deviations which occurs over a period of time.

Comparing the LSTM to the traditional techniques, both the stateful and stateless LSTM models perform better. The stateless LSTM is comparable to both the KNN and SVM model in how well it performed. This reinforces the idea that the stateless model did not in fact have the awareness of the current context for when predicting a class. The stateful LSTM is distinguished from other techniques in that it has a greater accuracy in predicting the overlapping areas of the scenarios. This stands in contrast to SVM and KNN where a measurement in the overlapping area, would result in a mostly random prediction. The mostly random decision making is indicated by the near 0.50 f1-score of class 0 in both the SVM and KNN models.

Further iterations, adjustments and tuning of the LSTM model could have resulted in a more accurate model. It is theorized, that adjustments to the dropout-layers would better help the model in better learn and maintain knowledge of previous experience. In doing so, it should be ensured that the resulting model is not overfit. The weakness of the multiclass classification model is the number of false positives it produces. False negatives, following the use case of this thesis can be accepted to a greater degree as long as the attack scenario is recognized at some point in the duration of the attack. The classification model, though now potentially being able to detect stealthy attacks, something that should be measured in an operational setting is the detection latency. In a real environment the number of false positives that are produced by the multiclass model would not be feasible to analyse. Another potential weakness of the developed classification models is the fact that the models are trained on scenarios with a fixed length. Each scenario is defined to be 1000 samples long in the simulated dataset. This might have introduce some bias into the model in that the model takes the length of the scenario into consideration, which is not ideal for a detection system which should rather use the values of the features. For future iterations of the simulated dataset, scenarios could be simulated with a randomized length for a defined interval.

## 5.2 Simulating dataset for intrusion detection

### 5.2.1 Simulating dataset for Critical Infrastructure

For this thesis, a methodology of using a simulated dataset to determine the capabilities of a ML model in detecting malicious behaviour was explored as an alternative to handling sensitive data. The methodology was tested in a use case, in an attempt to create a dataset which portrays relevant features of an electrical substation. The use case portrays a scenario where features used are assumed relevant and attack scenarios has a known and definable behaviour. The methodology proved to be a simple way of testing simple scenarios. Scenarios and environments can be defined and implemented quickly, depending on the scope and necessary complexity. By simulating the environment it is clearly defined what the trained model should be capable of identifying. Weaknesses of the model can therefore quickly be identified and understood by looking into what aspects of the given scenario was not taken into consideration by the classification model. Simulating the environment also has the advantage of being adaptable and scalable to changes which is wanted for the environment. Further scenarios, features and changes to existing scenarios and features can easily be made and adjusted. This is useful for testing the limits of the detection model and evaluating whether the model is capable of identifying new forms of attacks. In finding out that a current model does not recognize the new attack, it can simply be trained and evaluated on that particular scenario.

The challenge of entirely simulating a dataset is to build it accurately to the existing environment. A deep understanding of the environment and the relation between its features is needed for the results to be applicable. Large and complex environments are hardly possible to simulate manually. This goes for the environment of critical infrastructure, which is built on interconnected components that influence, control and communicate with each other. Defining such an environment, where features of a large number of components are interdependent, does neither optimize the efficiency nor result in an accurate resemblance. The dataset simulated as part of the thesis is deemed not to reflect the operations of the real-world environment as it was based on too few resources. There is still a lack of data portraying the behaviour of CI, even so its behaviour under different types of attacks.

Rather than attempting to simulate whole operations, a more feasible approach would be to simulate smaller instances of the environment or highly specific scenarios with a limited scope. Use of simulation can be incorporated together with real world data to train and test detection systems. Such an approach is especially needed in CI as there are few attack scenarios which can be performed towards the environment to generate data to be used in training and testing. Simulation can be incorporated to define behaviours of attacks which existing datasets does not portray, or extract data from unsafe conditions of the environment which otherwise would not possible to gather without risk.

### 5.2.2 Simulation in deep learning

As a result of the thesis, it has been recognized that creating a simulated dataset to develop and train a ML model, especially of deep architecture, does not play into the strengths of the neural network. Models such as the LSTM developed for this thesis, are especially efficient when working with a large amount of data. Deep models are capable of identify abstract and complex relations between features. In using a simulated dataset for a neural network to learn from, the complex relations are reduced to defined behaviour. In contrast, a trained model derived from the behaviour of real world data would be able to use the unidentified relations of the features to recognize and generalize a model which recognize the behaviour. When simulating features of a dataset the same abstract relations between the features are lost.

Deep learning also has the capability of extracting knowledge from data which has not been processed by feature engineering. Given the deep architecture, the model is capable of learning the complex structures of the raw data. The simulated dataset as applied in this thesis, assumes feature engineering has provided relevant features. The developed dataset therefore has a lower number of features. Compared to gathering real world data, features which are extracted, could be many, redundant and irrelevant. To best explore the capabilities of any given ML model, it would be a benefit to observe how it handles the magnitude of data which are found in the real environment. Generating a simulated dataset with similar number of features would take a lot of effort. In use of the simulated dataset to train a deep learning model, it leaves some of the potential on the table. As for the LSTM model which was developed for this thesis, the model is only trained on three features which means that there are still areas of unexplored potential of the model.

Analysis of the initial models, lead to clear answers to why the model did not perform optimally. This gave clear answers to what particular scenarios it is capable of finding and finds challenging to classify. The fact that the dataset is well defined, helps indicate why the model is not capable of performing the tasks it was created to do. Limitations which were found, were related to statefulness, model architecture and dataset structure. Knowing this, it becomes significantly easier to iterate and improve upon the shortcomings. Simulation has proven to be a useful tool in the development process, rather than an alternative to gather real world data from operational environment or testbeds. Simulation is an already frequently used technique in pre-processing of datasets, mostly to fill inn missing data. Expanding upon the idea, simulation could help build upon existing datasets to train and evaluate shortcomings of current classification models. With knowledge of the relations of the simulated features, it can be known whether or not the trained model recognizes that relation. The results of otherwise opaque models can be analysed and understood in regards to how they behave. From this both the dataset and the ML model can be adjusted and improved to better accommodate the requirements needed of the environment.

# 6 Conclusion

For the conclusion of this thesis the main findings and contribution of this thesis are presented. First, the research questions which was established in section 1.4 are answered. Later, the answers are expanded upon when discussing the contribution of the thesis. Thereafter, future work is presented, suggesting areas of research which could further explore usage of simulation in IDS for CI.

## 6.1 Research questions findings

The conclusion of the thesis will first be derived from the research questions which was established in the methodology.

1. **To what degree does a simulated dataset help indicate the efficiency of ML based IDS in CI?**

   It was from the performed experiments noted that an understanding in how the training data was derived gave a fundamental basis for understanding the trained model. Usage of a simulated dataset built on definable behaviour, gave clear guidelines for how the ML model was expected to behave. With knowledge of how the model should behave, weaknesses and missing traits of the model is easily identifiable. Understanding the dataset improved upon the transparency of the model. The developed LSTM model which was developed was shown to be transparent in regards to what scenarios it found difficult to classify, and what conditions would result in it being capable. The transparency of ML models through explainability benefits use of ML-based IDS in CI by understanding the limitations of the detection mechanism.

   The difficulty in simulating a dataset to portray CI is the complexity of ICS environments. It is not feasible for simulation alone to portray the complex behaviour and relations between components. The performance of a ML model which is tested on a fully simulated dataset, does therefore not fully reflect how it would operate in a real environment.

2. **To what degree does use of simulation in developing a dataset benefit the development of IDS in CI?**

   For the same reasons simulation can help indicating the performance of a ML model though transparency, transparency also enables development through understanding the behaviour of the model. Understanding why weaknesses of the model occur can help implement functions to improve the model. The initial LSTM model which was developed for the thesis, was found not to possess the necessary trait to classify the most challenging scenarios. The knowledge of how the scenarios was structured, indicated clearly what specific trait was needed to improve the model. This resulted in a new iteration of a better performing model, capable of classifying the previously challenging scenarios.

   Use of simulation in developing a dataset for CI also has the benefit of not being intrusive. Collection of data from CI, can be challenging as it is required to maintain operation. Simulation can be used as an alternative to generate data depicting scenarios which the dataset should include, without disruption. Simulation can help developing by both testing a current model on defined scenarios to see how it perform, or include such scenarios in a new model for a new iteration of a model to be capable of identifying it.

   In simulating the environment, it is however harder to gather the same number and the same complexity of features, than it is from a real environment. This is a disadvantage as a ML model, would make good use of the additional data for determining suspicious activity. Working with defined features reduces the potential complexity a ML model would be able to use for detection. What has made deep learning and AI-based classification models so useful in the area of IDS, is its ability to generalize complex relations and handle a large amount of features. It is therefore recognized that use of simulation does not play into all of the strengths of ML.

3. **Does a ML model built on a simulated dataset fulfill the requirements of, and benefit detection in the environment of CI?**

   Given the lack of relevant datasets which portrays the behaviour of ICS in CI, simulation is an optional method for acquiring relevant data. In comparison to using legacy datasets, simulation could be developed such that it better reflect relevant attack vectors and expected behaviour. However a challenge of simulating the environment is still the size and complexity. The transparency which the simulated dataset brought to the ML model, was shown to be of benefit to the CI which was simulated. Greater accuracy was achieved in understanding how to improve upon the developed model. The developed stateful LSTM classification model proposed, show potential use in ICS environment, due to its capability in detecting attacks which only affect measured features by small margins. The model is capable of using previous context in determining the current state of the system, enabling it in detecting abnormalities which can be measured for over periods of time. This could be useful for detecting stealthy attacks which in CI, could cause costly damages.

## 6.2 Contribution

Due to the complexity which an ICS might possess, it is deemed unfeasible and therefore not beneficial to manually simulate its behaviour. The interconnected components lead to an environment which is hard to represent by defined scenarios. However, simulation could be a useful tool in learning how an otherwise opaque model behaves and enhancing its capabilities. Usage of simulation in combination with relevant dataset to generate otherwise unseen behaviour, can be both used to evaluate and train the model on that type of behaviour. Given the challenges and regulations needed to collect and use data from CI, simulation could be a useful method to fulfill and expand upon a dataset with scenarios otherwise not possible to extract from the operational environment or testbed. In the case of gathering data of a testbed, simulation could be used to better adjust the dataset to the actual operational environment by including additional features of components or protocols. When developing a model for generalizing the behaviour of ICS, noise and missing data should also be introduced, as simulation in contrast to real operation does not possess the same types of imperfections as might be seen in operation.

Another reason for why it would be beneficial to include usage of real-world data, is that a simulated dataset does not play into the strength of deep learning. DNN's are great at using large amount of features and data to find complex relations amongst features. When creating a dataset which is built on simulated features, the complex relations are reduced to defined values. Some of the benefits in using DNN are thereby lost until a certain level of complexity, as found in real-world operation, is introduced to the dataset. This can be accomplished by using datasets derived from real operation, or finding alternative methods of simulation which accurately mimics an environments behaviour.

From the established use case for the thesis, the resulting ML model became more transparent in how it operated when built on a dataset which was defined. This makes the results of evaluation more efficient in that observing what simulated rules was not recognized, clearly indicate what aspects of the model should be further developed upon. The LSTM model which was developed, was initially not capable of identifying the most challenging scenarios. It was from the analysis deducted that the model did not possess the trait of identifying the variety of previous samples in identifying the current class. This knowledge was used to improve the training model to be capable of distinguishing the classes more precisely. The new iteration of the model was implemented as stateful, which made it aware of previously measured values of previous batches, in determining the current behaviour. The outcome of developing the model on a simulated dataset, built on defined scenarios, gave a clear understanding of what traits the model was missing and how to further develop and improve the model. This gives a major advantage compared to working with traditional datasets, where the capabilities and traits of an opaque model might not necessarily be extracted from testing.

In CI, use of ML should be used in such a way that the workings of how a model operates, can be explained. Especially in the case of deep learning, the higher complexity and dimensionality makes it hard to understand how the model functions. In classification, this means that why a particular sample was classified the way it was, is not clear to an observer. The model might therefore be making predictions based on a false understanding of the environment, without any way for an observer to see where it went wrong. Though the methodology of a defined simulation has benefited the explainability, further work is required to enhance the explainability of models created on operational datasets, and ensure the security of CI.

## 6.3 Future work

In the area of CI, there is still a lack of relevant datasets which portray the behaviour of ICS. This has made it challenging relating research of proposed ML models as many of them are trained and evaluated on mostly irrelevant and outdated datasets. Further work is required to generate and find solutions to best support development of detection methodologies in the CI environment. These methodologies and proposed models should also support a high level of explainability to ensure that results of the evaluated performance are applicable to the real environment. The transparency of models are especially important for CI to better understand how the model operates and thereby understand its weaknesses.

Use of simulation is a tool which has the potential of benefiting the development of ML models. It is therefore needed a greater background of simulation methodologies which are compared to real operation. It is required a greater understanding of what information is lost when simulating an environment instead of gathering real-world measurements. Further development of tools and programs to virtually simulate these complex environments is also needed to accurately mimic the real systems found in ICS.

# References

[1] A. Pinto, L. C. Herrera, Y. Donoso and J. A. Gutierrez, 'Survey on intrusion detection systems based on machine learning techniques for the protection of critical infrastructure', *Sensors (Basel, Switzerland)*, vol. 23, 5 Mar. 2023, ISSN: 14248220. DOI: 10.3390/S23052415.

[2] M. R. G. Raman, N. Somu and A. P. Mathur, 'Anomaly detection in critical infrastructure using probabilistic neural network', *Communications in Computer and Information Science*, vol. 1116 CCIS, pp. 129–141, 2019, ISSN: 18650937. DOI: 10.1007/978-981-15-0871-4_10/TABLES/3. [Online]. Available: https://link.springer.com/chapter/10.1007/978-981-15-0871-4_10.

[3] W. Xu, Y. Tao and X. Guan, 'The landscape of industrial control systems (ics) devices on the internet', pp. 1–8, 2018. DOI: 10.1109/CyberSA.2018.8551422.

[4] S. Soliman, W. Oudah and A. Aljuhani, 'Deep learning-based intrusion detection approach for securing industrial internet of things', *Alexandria Engineering Journal*, vol. 81, pp. 371–383, Oct. 2023, ISSN: 1110-0168. DOI: 10.1016/J.AEJ.2023.09.023.

[5] E. Anthi, L. Williams, M. Rhode, P. Burnap and A. Wedgbury, 'Adversarial attacks on machine learning cybersecurity defences in industrial control systems', *Journal of Information Security and Applications*, vol. 58, p. 102 717, May 2021, ISSN: 2214-2126. DOI: 10.1016/J.JISA.2020.102717.

[6] J. Kirupakar and S. M. Shalinie, 'Situation aware intrusion detection system design for industrial iot gateways', *ICCIDS 2019 - 2nd International Conference on Computational Intelligence in Data Science, Proceedings*, Feb. 2019. DOI: 10.1109/ICCIDS.2019.8862038.

[7] M. Liu, Z. Xue, X. Xu, C. Zhong and J. Chen, 'Host-based intrusion detection system with system calls', *ACM Computing Surveys (CSUR)*, vol. 51, 5 Nov. 2018, ISSN: 15577341. DOI: 10.1145/3214304. [Online]. Available: https://dl.acm.org/doi/10.1145/3214304.

[8] J. O. Storeng, 'Deep learning approach for host-based intrusion detection system in critical infrastructure', Department of Information Security, Communication NTNU – Norwegian University of Science and Technology, Project report in IMT4205 - Research Project Planning, Dec. 2023.

[9] F. Panagiotis, K. Taxiarxchis, K. Georgios, L. Maglaras and M. A. Ferrag, 'Intrusion detection in critical infrastructures: A literature review', *Smart Cities 2021, Vol. 4, Pages 1146-1157*, vol. 4, pp. 1146–1157, 3 Aug. 2021, ISSN: 2624-6511. DOI: 10.3390/SMARTCITIES4030061. [Online]. Available: https://www.mdpi.com/2624-6511/4/3/61/htm%20https://www.mdpi.com/2624-6511/4/3/61.

[10] D. Bhamare, M. Zolanvari, A. Erbad, R. Jain, K. Khan and N. Meskin, 'Cybersecurity for industrial control systems: A survey', *Computers & Security*, vol. 89, p. 101 677, Feb. 2020, ISSN: 0167-4048. DOI: 10.1016/J.COSE.2019.101677.

[11] A. Al-Abassi, H. Karimipour, A. Dehghantanha and R. M. Parizi, 'An ensemble deep learning-based cyber-attack detection in industrial control system', *IEEE Access*, vol. 8, pp. 83 965–83 973, 2020, ISSN: 21693536. DOI: 10.1109/ACCESS.2020.2992249.

[12] M. Begli, F. Derakhshan and H. Karimipour, 'A layered intrusion detection system for critical infrastructure using machine learning', *Proceedings of 2019 the 7th International Conference on Smart Energy Grid Engineering, SEGE 2019*, pp. 120–124, Aug. 2019. DOI: 10.1109/SEGE.2019.8859950.

[13] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, A. Al-Nemrat and S. Venkatraman, 'Deep learning approach for intelligent intrusion detection system', *IEEE Access*, vol. 7, pp. 41 525–41 550, 2019. DOI: 10.1109/ACCESS.2019.2895334.

[14] H. Yang, L. Cheng and M. C. Chuah, 'Deep-learning-based network intrusion detection for scada systems', *2019 IEEE Conference on Communications and Network Security, CNS 2019*, Jun. 2019. DOI: 10.1109/CNS.2019.8802785.

[15] Á. L. P. Gómez, L. F. Maimó, A. H. Celdrán *et al.*, 'On the generation of anomaly detection datasets in industrial control systems', *IEEE Access*, vol. 7, pp. 177 460–177 473, 2019, ISSN: 21693536. DOI: 10.1109/ACCESS.2019.2958284.

[16] F. Laghrissi, S. Douzi, K. Douzi and B. Hssina, 'Intrusion detection systems using long short-term memory (lstm)', *Journal of Big Data*, vol. 8, no. 1, p. 65, 2021.

[17] U. K. Premaratne, J. Samarabandu, T. S. Sidhu, R. Beresh and J.-C. Tan, 'An intrusion detection system for iec61850 automated substations', *IEEE Transactions on Power Delivery*, vol. 25, no. 4, pp. 2376–2383, 2010. DOI: 10.1109/TPWRD.2010.2050076.

[18] M. Dawson, R. Bacius, L. B. Gouveia and A. Vassilakos, 'Understanding the challenge of cybersecurity in critical infrastructure sectors', *Land Forces Academy Review*, vol. 26, no. 1, pp. 69–75, 2021.

[19] M. Conti, D. Donadel and F. Turrin, 'A survey on industrial control system testbeds and datasets for security research', *IEEE Communications Surveys and Tutorials*, vol. 23, pp. 2248–2294, 4 2021, ISSN: 1553877X. DOI: 10.1109/COMST.2021.3094360.

[20] C. Ekisa, D. Ó. Briain and Y. Kavanagh, 'An open-source testbed to visualise ics cybersecurity weaknesses and remediation strategies – a research agenda proposal', pp. 1–6, 2021. DOI: 10.1109/ISSC52156.2021.9467852.

[21] M. Ghobakhloo, 'Industry 4.0, digitization, and opportunities for sustainability', *Journal of cleaner production*, vol. 252, p. 119 869, 2020.

[22] T. Zheng, M. Ardolino, A. Bacchetti and M. Perona, 'The applications of industry 4.0 technologies in manufacturing context: A systematic literature review', *International Journal of Production Research*, vol. 59, no. 6, pp. 1922–1954, 2021.

[23] I. Ali, A. I. A. Ahmed, A. Almogren *et al.*, 'Systematic literature review on iot-based botnet attack', *IEEE access*, vol. 8, pp. 212 220–212 232, 2020.

[24] M. binti Mohamad Noor and W. H. Hassan, 'Current research on internet of things (iot) security: A survey', *Computer Networks*, vol. 148, pp. 283–294, 2019, ISSN: 1389-1286. DOI: https://doi.org/10.1016/j.comnet.2018.11.025. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128618307035.

[25] L. Piètre-Cambacédès, M. Tritschler and G. N. Ericsson, 'Cybersecurity myths on power control systems: 21 misconceptions and false beliefs', *IEEE Transactions on Power Delivery*, vol. 26, no. 1, pp. 161–172, 2011. DOI: 10.1109/TPWRD.2010.2061872.

[26] L. Burita and D. T. Le, 'Cyber security and apt groups', pp. 1–7, 2021. DOI: 10.1109/KIT52904.2021.9583744.

[27] L. Bilge and T. Dumitraş, 'Before we knew it: An empirical study of zero-day attacks in the real world', in *Proceedings of the 2012 ACM conference on Computer and communications security*, 2012, pp. 833–844.

[28] S. N. Thanh Vu, M. Stege, P. I. El-Habr, J. Bang and N. Dragoni, 'A survey on botnets: Incentives, evolution, detection and current trends', *Future Internet*, vol. 13, no. 8, p. 198, 2021.

[29] X. Zhang, O. Upton, N. L. Beebe and K.-K. R. Choo, 'Iot botnet forensics: A comprehensive digital forensic case study on mirai botnet servers', *Forensic Science International: Digital Investigation*, vol. 32, p. 300 926, 2020.

[30] A. Affinito, S. Zinno, G. Stanco, A. Botta and G. Ventre, 'The evolution of mirai botnet scans over a six-year period', *Journal of Information Security and Applications*, vol. 79, p. 103 629, 2023.

[31] M. Wazzan, D. Algazzawi, O. Bamasaq, A. Albeshri and L. Cheng, 'Internet of things botnet detection approaches: Analysis and recommendations for future research', *Applied Sciences*, vol. 11, no. 12, p. 5713, 2021.

[32] S. Lee, A. Abdullah, N. Jhanjhi, K. S. Hoong and S. Jaya, 'Classification of botnet attacks in iot smart factory',

[33] M. Baezner and P. Robin, 'Stuxnet', ETH Zurich, Tech. Rep., 2017.

[34] P. Kozak, I. Klaban and T. Šlajs, 'Industroyer cyber-attacks on ukraine's critical infrastructure', in *2023 International Conference on Military Technologies (ICMT)*, IEEE, 2023, pp. 1–6.

[35] A. Akbarzadeh, L. Erdodi, S. H. Houmb, T. G. Soltvedt and H. K. Muggerud, 'Attacking iec 61850 substations by targeting the ptp protocol', *Electronics*, vol. 12, no. 12, 2023, ISSN: 2079-9292. DOI: 10.3390/electronics12122596. [Online]. Available: https://www.mdpi.com/2079-9292/12/12/2596.

[36] L. Erdődi, P. Kaliyar, S. H. Houmb, A. Akbarzadeh and A. J. Waltoft-Olsen, 'Attacking power grid substations: An experiment demonstrating how to attack the scada protocol iec 60870-5-104', in *Proceedings of the 17th International Conference on Availability, Reliability and Security*, ser. ARES '22, Vienna, Austria: Association for Computing Machinery, 2022, ISBN: 9781450396707. DOI: 10.1145/3538969.3544475. [Online]. Available: https://doi.org/10.1145/3538969.3544475.

[37] J.-M. Storm, S. H. Houmb, P. Kaliyar, L. Erdodi and J. M. Hagen, 'Testing commercial intrusion detection systems for industrial control systems in a substation hardware in the loop testlab', *Electronics*, vol. 13, no. 1, 2024, ISSN: 2079-9292. DOI: 10.3390/electronics13010060. [Online]. Available: https://www.mdpi.com/2079-9292/13/1/60.

[38] W. Aoudi, M. Iturbe and M. Almgren, 'Truth will out: Departure-based process-level detection of stealthy attacks on control systems', in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018, pp. 817–831.

[39] N. Sharma, R. Sharma and N. Jindal, 'Machine learning and deep learning applications-a vision', *Global Transitions Proceedings*, vol. 2, no. 1, pp. 24–28, 2021.

[40] I. Kononenko and M. Kukar, *Machine learning and data mining*. Horwood Publishing, 2007.

[41] K. Sharifani and M. Amini, 'Machine learning and deep learning: A review of methods and applications', *World Information Technology and Engineering Journal*, vol. 10, no. 07, pp. 3897–3904, 2023.

[42] H. U. Dike, Y. Zhou, K. K. Deveerasetty and Q. Wu, 'Unsupervised learning based on artificial neural network: A review', pp. 322–327, 2018. DOI: 10.1109/CBS.2018.8612259.

[43] P. Parkar and A. Bilimoria, 'A survey on cyber security ids using ml methods', in *2021 5th International Conference on Intelligent Computing and Control Systems (ICICCS)*, IEEE, 2021, pp. 352–360.

[44] F. Khushaktov, *Introduction random forest classification by example*, Aug. 2023. [Online]. Available: https://medium.com/@mrmaster907/introduction-random-forest-classification-by-example-6983d95c7b91.

[45] H. Bhavsar and M. H. Panchal, 'A review on support vector machine for data classification', *International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)*, vol. 1, no. 10, pp. 185–189, 2012.

[46] A. Krenker, J. Bešter and A. Kos, 'Introduction to the artificial neural networks', *Artificial Neural Networks: Methodological Advances and Biomedical Applications. InTech*, pp. 1–18, 2011.

[47] C. Sekhar and P. S. Meghana, 'A study on backpropagation in artificial neural networks', *Asia-Pacific Journal of Neural Networks and Its Applications*, vol. 4, no. 1, pp. 21–28, 2020.

[48] A. Aldweesh, A. Derhab and A. Z. Emam, 'Deep learning approaches for anomaly-based intrusion detection systems: A survey, taxonomy, and open issues', *Knowledge-Based Systems*, vol. 189, p. 105 124, Feb. 2020, ISSN: 0950-7051. DOI: 10.1016/J.KNOSYS.2019.105124.

[49] A. Sherstinsky, 'Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network', *Physica D: Nonlinear Phenomena*, vol. 404, p. 132 306, 2020.

[50] A. Saxena, *Introduction to long short-term memory (lstm)*, Jan. 2023. [Online]. Available: https://medium.com/analytics-vidhya/introduction-to-long-short-term-memory-lstm-a8052cd0d4cd.

[51] K. Smagulova and A. P. James, 'A survey on lstm memristive neural network architectures and applications', *The European Physical Journal Special Topics*, vol. 228, no. 10, pp. 2313–2324, 2019.

[52] C. Olah, *Understanding lstm networks*, Aug. 2015. [Online]. Available: https://colah.github.io/posts/2015-08-Understanding-LSTMs/.

[53] I. Bismi, *Difference between stateful and stateless rnns*, May 2023. [Online]. Available: https://medium.com/@iqra.bismi/difference-between-stateful-and-stateless-rnns-2b397184e759.

[54] A. Katrompas and V. Metsis, 'Enhancing lstm models with self-attention and stateful training', in *Intelligent Systems and Applications: Proceedings of the 2021 Intelligent Systems Conference (IntelliSys) Volume 1*, Springer, 2022, pp. 217–235.

[55] A. A. Awan, *The curse of dimensionality in machine learning: Challenges, impacts, and solutions*, Accessed on Feb 26, 2024, 2023. [Online]. Available: https://www.datacamp.com/blog/curse-of-dimensionality-machine-learning.

[56] A. Alazab, M. Hobbs, J. Abawajy and M. Alazab, 'Using feature selection for intrusion detection system', pp. 296–301, 2012.

[57] J. Cai, J. Luo, S. Wang and S. Yang, 'Feature selection in machine learning: A new perspective', *Neurocomputing*, vol. 300, pp. 70–79, 2018.

[58] D. Minh, ·. H. X. Wang, ·. Y. F. Li and T. N. Nguyen, 'Explainable artificial intelligence: A comprehensive review', *Artificial Intelligence Review*, vol. 55, pp. 3503–3568, 2022. DOI: 10.1007/s10462-021-10088-y. [Online]. Available: https://doi.org/10.1007/s10462-021-10088-y.

[59] P. Besse, C. Castets-Renard, A. Garivier and J.-M. Loubes, 'Can everyday ai be ethical? machine learning algorithm fairness (english version)', Nov. 2018. DOI: 10.13140/RG.2.2.22973.31207.

[60] A. Adadi and M. Berrada, 'Peeking inside the black-box: A survey on explainable artificial intelligence (xai)', *IEEE Access*, vol. 6, pp. 52 138–52 160, 2018. DOI: 10.1109/ACCESS.2018.2870052.

[61] D. Gunning, M. Stefik, J. Choi, T. Miller, S. Stumpf and G. Z. Yang, 'Xai—explainable artificial intelligence', *Science Robotics*, vol. 4, 37 Dec. 2019, ISSN: 24709476. DOI: 10.1126/SCIROBOTICS.AAY7120. [Online]. Available: https://www.science.org/doi/10.1126/scirobotics.aay7120.

[62] S. V. Rakas, M. D. Stojanovic and J. D. Markovic-Petrovic, 'A review of research work on network-based scada intrusion detection systems', *IEEE Access*, vol. 8, pp. 93 083–93 108, 2020, ISSN: 21693536. DOI: 10.1109/ACCESS.2020.2994961.

[63] Z. Ahmad, A. S. Khan, C. W. Shiang, J. Abdullah and F. Ahmad, 'Network intrusion detection system: A systematic study of machine learning and deep learning approaches', *Transactions on Emerging Telecommunications Technologies*, vol. 32, 1 Jan. 2021, ISSN: 21613915. DOI: 10.1002/ETT.4150.

[64] H.-J. Liao, C.-H. Richard Lin, Y.-C. Lin and K.-Y. Tung, 'Intrusion detection system: A comprehensive review', *Journal of Network and Computer Applications*, vol. 36, no. 1, pp. 16–24, 2013, ISSN: 1084-8045. DOI: https://doi.org/10.1016/j.jnca.2012.09.004. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1084804512001944.

[65] Z. Ahmad, A. Shahid Khan, C. Wai Shiang, J. Abdullah and F. Ahmad, 'Network intrusion detection system: A systematic study of machine learning and deep learning approaches', *Transactions on Emerging Telecommunications Technologies*, vol. 32, no. 1, e4150, 2021.

[66] V. S. Rajkumar, M. Tealane, A. Ştefanov, A. Presekal and P. Palensky, 'Cyber attacks on power system automation and protection and impact analysis', pp. 247–254, 2020. DOI: 10.1109/ISGT-Europe47291.2020.9248840.

[67] T. O. Olowu, S. Dharmasena, A. Hernandez and A. Sarwat, 'Impact analysis of cyber attacks on smart grid: A review and case study', *New Research Directions in Solar Energy Technologies*, pp. 31–51, 2021.

[68] V. S. Rajkumar, M. Tealane, A. Ştefanov and P. Palensky, 'Cyber attacks on protective relays in digital substations and impact analysis', pp. 1–6, 2020. DOI: 10.1109/MSCPES49613.2020.9133698.

# A Appendix

## A.1 Acronyms and abbreviations

## Acronyms

**NIDS** Network-based Intrusion Detection System. v, 5, 18, 19

**OT** Operational Technology. 10, 12

**PLC** Programmable Logic Controller. 8

**PNN** Probabilistic Neural Network. 3

**RNN** Recurrent Neural Network. v, vii, 14–16, 25

**SCADA** Supervisory Control and Data Acquisition. 3, 5, 8, 12

**SIDS** Signature-based Intrusion Detection System. v, 4, 18

**SIS** Safety Instrumentation Systems. 8

**SPA** Stateful Protocol Analysis. v, 18

**SVM** Support Vector Machine. vi–viii, 4, 14, 32, 33, 36, 40, 41, 65

**TN** True Negative. 28

**TP** True Positive. 28

**TPR** True Positive Rate. 28

**XAI** Explainable artificial intelligence. v, 7, 17

## A.2 Code

### A.2.1 Simulating muliclass dataset

```python
import numpy as np
import pandas as pd
import random

def getMeasure(prev, avg, var):
  if prev < (avg + var) and prev > (avg - var):
    if random.randrange(0,10) > 9:
        return avg
    else:
      return round(avg + random.uniform(-var, var), 3)
  else:
    return avg

def getSamples(samples, man_freq_Avg, man_freq_var, man_volt_Avg, man_volt_var,
    man_curr_Avg, man_curr_var, is_attack, label):
  freq_Avg = 50.0
  freq_var = 0.02

  volt_Avg = 400.0
  volt_var = 2.0

  curr_Avg = 150.0
  curr_var = 1.0

  malarr = [[getMeasure(50.0, freq_Avg, freq_var), getMeasure(400, volt_Avg,
    volt_var), getMeasure(150, curr_Avg, curr_var),0 ,0]]
  for i in range(samples):
    newArr = [[getMeasure(malarr[i][0], man_freq_Avg, man_freq_var), getMeasure(
    malarr[i][1], man_volt_Avg, man_volt_var), getMeasure(malarr[i][2],
    man_curr_Avg, man_curr_var), is_attack, label]]
    malarr = np.append(malarr, newArr, axis=0)
  return malarr

nr_attacks = 6

meas_val = [[50.0, 0.02, 400.0, 2.0, 150.0, 1.0, 0, 0,], #normal behaviour
            [50, 0.05, 200, 50, 100, 10, 1, 1], #high impact DoS
            [50, 0.02, 100, 10, 150, 1, 1, 2], #High impact process manipulation
            [50.02, 0.01, 100, 10, 180, 1, 1, 3], #Medium impact process
    manipulation
            [50.0, 0.02, 500, 20, 200, 5, 1, 4], #Medium impact replay attack
            [50.0, 0.01, 400, 1, 150, 0.8, 1, 5], #Low impact stealth attack
            [50.0, 0.03, 400, 3, 150, 1.5, 1, 6]] #Low impact direct damage attack


samples_per_scenario = 1000
train_nr_iterations = 1000
test_nr_iterations = 200


'''train'''
train = getSamples(samples_per_scenario, meas_val[0][0], meas_val[0][1], meas_val
    [0][2], meas_val[0][3], meas_val[0][4], meas_val[0][5], meas_val[0][6],
    meas_val[0][7])
for i in range(train_nr_iterations):
  np.random.shuffle(meas_val)
  for j in range(nr_attacks + 1):
    train = np.append(train, getSamples(samples_per_scenario, meas_val[j][0],
    meas_val[j][1], meas_val[j][2], meas_val[j][3], meas_val[j][4], meas_val[j][5],
     meas_val[j][6], meas_val[j][7]), axis=0)

'''test'''
test = getSamples(samples_per_scenario, meas_val[0][0], meas_val[0][1], meas_val
    [0][2], meas_val[0][3], meas_val[0][4], meas_val[0][5], meas_val[0][6],
    meas_val[0][7])
for i in range(test_nr_iterations):
  np.random.shuffle(meas_val)
  for j in range(nr_attacks + 1):
    test = np.append(test, getSamples(samples_per_scenario, meas_val[j][0], meas_val[j][1], meas_val[j][2], meas_val[j][3], meas_val[j][4], meas_val[j][5], meas_val[j][6], meas_val[j][7]), axis=0)

trainSet = pd.DataFrame(data=train[0:, 0:], columns=['frequency', 'volt', 'current','is_attack', 'label'])
testSet = pd.DataFrame(data=test[0:, 0:], columns=['frequency', 'volt', 'current', 'is_attack', 'label'])

trainSet.to_csv('train_simulated.csv')
testSet.to_csv('test_simulated.csv')
```

### A.2.2 Simulating binary dataset

```python
import numpy as np
import pandas as pd
import random

def getMeasure(prev, avg, var):
  if prev < (avg + var) and prev > (avg - var):
    if random.randrange(0,10) > 9:
        return avg
    else:
      return round(avg + random.uniform(-var, var), 3)
  else:
    return avg

def getSamples(samples, man_freq_Avg, man_freq_var, man_volt_Avg, man_volt_var, man_curr_Avg, man_curr_var, is_attack, label):
  freq_Avg = 50.0
  freq_var = 0.02

  volt_Avg = 400.0
  volt_var = 2.0

  curr_Avg = 150.0
  curr_var = 1.0

  malarr = [[getMeasure(50.0, freq_Avg, freq_var), getMeasure(400, volt_Avg, volt_var), getMeasure(150, curr_Avg, curr_var),0 ,0]]
  for i in range(samples):
    newArr = [[getMeasure(malarr[i][0], man_freq_Avg, man_freq_var), getMeasure(malarr[i][1], man_volt_Avg, man_volt_var), getMeasure(malarr[i][2], man_curr_Avg, man_curr_var), is_attack, label]]
    malarr = np.append(malarr, newArr, axis=0)
  return malarr

nr_attacks = 6

meas_val = [[50.0, 0.02, 400.0, 2.0, 150.0, 1.0, 0, 0,], #normal behaviour
            [50.0, 0.02, 400.0, 2.0, 150.0, 1.0, 0, 0,], #normal behaviour
      [50.0, 0.02, 400.0, 2.0, 150.0, 1.0, 0, 0,], #normal behaviour
      [50.0, 0.02, 400.0, 2.0, 150.0, 1.0, 0, 0,], #normal behaviour
      [50.0, 0.02, 400.0, 2.0, 150.0, 1.0, 0, 0,], #normal behaviour
      [50.0, 0.02, 400.0, 2.0, 150.0, 1.0, 0, 0,], #normal behaviour
      [50, 0.05, 200, 50, 100, 10, 1, 1], #high impact DoS
            [50, 0.02, 100, 10, 150, 1, 1, 2], #High impact process manipulation
            [50.02, 0.01, 100, 10, 180, 1, 1, 3], #Medium impact process manipulation
            [50.0, 0.02, 500, 20, 200, 5, 1, 4], #Medium impact replay attack
            [50.0, 0.01, 400, 1, 150, 0.8, 1, 5], #Low impact stealth attack
            [50.0, 0.03, 400, 3, 150, 1.5, 1, 6]] #Low impact direct damage attack


samples_per_scenario = 1000
train_nr_iterations = 1000
test_nr_iterations = 200

print(range(len(meas_val)))

'''train'''
train = getSamples(samples_per_scenario, meas_val[0][0], meas_val[0][1], meas_val[0][2], meas_val[0][3], meas_val[0][4], meas_val[0][5], meas_val[0][6], meas_val[0][7])
for i in range(train_nr_iterations):
  np.random.shuffle(meas_val)
  for j in range(len(meas_val)):
    train = np.append(train, getSamples(samples_per_scenario, meas_val[j][0], meas_val[j][1], meas_val[j][2], meas_val[j][3], meas_val[j][4], meas_val[j][5], meas_val[j][6], meas_val[j][7]), axis=0)

'''test'''
test = getSamples(samples_per_scenario, meas_val[0][0], meas_val[0][1], meas_val[0][2], meas_val[0][3], meas_val[0][4], meas_val[0][5], meas_val[0][6], meas_val[0][7])
```

```
for i in range(test_nr_iterations):
  np.random.shuffle(meas_val)
  for j in range(len(meas_val)):
    test = np.append(test, getSamples(samples_per_scenario, meas_val[j][0],
    meas_val[j][1], meas_val[j][2], meas_val[j][3], meas_val[j][4], meas_val[j][5],
     meas_val[j][6], meas_val[j][7]), axis=0)

trainSet = pd.DataFrame(data=train[0:, 0:], columns=['frequency', 'volt', 'current'
    ,'is_attack', 'label'])
testSet = pd.DataFrame(data=test[0:, 0:], columns=['frequency', 'volt', 'current',
    'is_attack', 'label'])

trainSet.to_csv('train_simulated_binary.csv')
testSet.to_csv('test_simulated_binary.csv')
```

### A.2.3 Train Multiclass LSTM model

```python

import numpy as np
import matplotlib.pyplot as plt
from keras.models import Sequential
from keras.layers import Dense, LSTM
import tensorflow as tf
import pandas as pd
from sklearn import preprocessing
from sklearn.preprocessing import LabelEncoder
from keras.utils import to_categorical
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, InputLayer

n_features = 3

def get_Model(data, batch_size):
  timesteps = data.shape[1]
  features = data.shape[2]

  model = Sequential()
  model.add(InputLayer(batch_input_shape=(batch_size, timesteps, features)))
  model.add(LSTM(64, return_sequences=True, stateful=False))
  model.add(Dropout(0.02))
  model.add(LSTM(32, return_sequences=True, stateful=False))
  model.add(Dropout(0.02))
  model.add(LSTM(16, return_sequences=False, stateful=True))
  model.add(Dropout(0.02))
  model.add(Dense(7, activation='softmax'))
  return model

def generate_seq(data):
  seqs=[]
  for i in range(0, len(data) - seq_length + 1, seq_overlap):
    seqs.append(data.iloc[i:i+seq_length, :])
  return seqs

def pad_sequences(seqs):
    padded_seqs = []
    for seq in seqs:
        if len(seq) < seq_length:
            padded_seq = np.concatenate((seq, np.zeros((seq_length-len(seq), len(df
    .columns)))), axis=0)
        else:
            padded_seq = seq
        padded_seqs.append(padded_seq)
    return np.array(padded_seqs)


df_train = pd.read_csv("train_simulated.csv", index_col=0)
df_test = pd.read_csv("test_simulated.csv", index_col=0)
print(df_train.head())

X_train = df_train[['frequency', 'volt', 'current']]
Y_train = df_train['label']

X_test = df_test[['frequency', 'volt', 'current']]
Y_test = df_test['label']

min_max_scaler = preprocessing.MinMaxScaler()
X_train = pd.DataFrame(min_max_scaler.fit_transform(X_train.values), columns=
    X_train.columns)
X_test = pd.DataFrame(min_max_scaler.fit_transform(X_test.values), columns=X_test.
    columns)

X_train = X_train.to_numpy()
X_test = X_test.to_numpy()

X_train = X_train.reshape(X_train.shape[0], 1, X_train.shape[1])
X_test = X_test.reshape(X_test.shape[0], 1, X_test.shape[1])

print(np.unique(Y_train, axis=0))
```

```

encoder = LabelEncoder()
Y_train_fit = encoder.fit_transform(Y_train)
Y_test_fit = encoder.fit_transform(Y_train)
encoder.fit(Y_train_fit)
encoder.fit(Y_test_fit)
Y_train_enc = encoder.transform(Y_train)
Y_test_enc = encoder.transform(Y_test)
Y_train = to_categorical(Y_train_enc, num_classes=7)
Y_test = to_categorical(Y_test_enc, num_classes=7)

print(X_train.shape, Y_train.shape)

batch_size = 128
epochs = 25

model = get_Model(X_train, batch_size)
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy
    '])

num_samples = len(X_train)
print(num_samples)
if num_samples % batch_size != 0:
    num_samples = (num_samples // batch_size) * batch_size
    X_train = X_train[:num_samples]
    Y_train = Y_train[:num_samples]

history = model.fit(X_train, Y_train, batch_size=batch_size, epochs=epochs, shuffle
    =False, validation_split=0.2)
print(model.summary())

from sklearn.metrics import (precision_score, recall_score, f1_score,
    accuracy_score, mean_squared_error, mean_absolute_error)
from sklearn.metrics import confusion_matrix
from sklearn.metrics import multilabel_confusion_matrix

num_samples = len(X_test)
print(num_samples)
if num_samples % batch_size != 0:
    num_samples = (num_samples // batch_size) * batch_size
    X_test = X_test[:num_samples]
    Y_test = Y_test[:num_samples]

results = model.evaluate(X_test, Y_test, batch_size=batch_size)
print(results)
print("Accuracy: %.2f%%" % (results[1]*100))

model.save('LSTM_model.keras')

plt.figure(1)
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Predicted and real loss value')
plt.xlabel('Epoch')
plt.ylabel('Loss persentage')
plt.legend(['Predict_loss','Real_loss'])
plt.show()
plt.savefig("loss_plt.png")

plt.figure(2)
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('prediction and real accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy persentage')
plt.legend(['Predicted_Accuracy', 'Real_Accuracy'])
plt.show()
plt.savefig("Acc_plt.png")
```

### A.2.4 Train Binary LSTM model

```python
import numpy as np
import matplotlib.pyplot as plt
from keras.models import Sequential
from keras.layers import Dense, LSTM
import tensorflow as tf
import pandas as pd
from sklearn import preprocessing
from sklearn.preprocessing import LabelEncoder
from keras.utils import to_categorical
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, InputLayer

def get_Model(data, batch_size):
  timesteps = data.shape[1]
  features = data.shape[2]

  model = Sequential()
  model.add(InputLayer(batch_input_shape=(batch_size, timesteps, features)))
  model.add(LSTM(64, return_sequences=True, stateful=False))
  model.add(Dropout(0.02))
  model.add(LSTM(32, return_sequences=True, stateful=False))
  model.add(Dropout(0.02))
  model.add(LSTM(16, return_sequences=False, stateful=True))
  model.add(Dropout(0.02))
  model.add(Dense(2, activation='sigmoid'))
  return model

df_train = pd.read_csv("train_simulated_binary.csv", index_col=0)
df_test = pd.read_csv("test_simulated_binary.csv", index_col=0)
print(df_train.head())

X_train = df_train[['frequency', 'volt', 'current']]
Y_train = df_train['is_attack']

X_test = df_test[['frequency', 'volt', 'current']]
Y_test = df_test['is_attack']

min_max_scaler = preprocessing.MinMaxScaler()
X_train = pd.DataFrame(min_max_scaler.fit_transform(X_train.values), columns=
    X_train.columns)
X_test = pd.DataFrame(min_max_scaler.fit_transform(X_test.values), columns=X_test.
    columns)

X_train = X_train.to_numpy()
X_test = X_test.to_numpy()

X_train = X_train.reshape(X_train.shape[0], 1, X_train.shape[1])
X_test = X_test.reshape(X_test.shape[0], 1, X_test.shape[1])

print(np.unique(Y_train, axis=0))

encoder = LabelEncoder()
Y_train_fit = encoder.fit_transform(Y_train)
Y_test_fit = encoder.fit_transform(Y_train)
encoder.fit(Y_train_fit)
encoder.fit(Y_test_fit)
Y_train_enc = encoder.transform(Y_train)
Y_test_enc = encoder.transform(Y_test)
Y_train = to_categorical(Y_train_enc, num_classes=2)
Y_test = to_categorical(Y_test_enc, num_classes=2)

print(X_train.shape, Y_train.shape)

batch_size = 128
epochs = 25

model = get_Model(X_train, batch_size)
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])


#validation split 0.2
```

```python
X_val = X_train[round(len(X_train)*0.8):]
Y_val = Y_train[round(len(X_train)*0.8):]
X_train = X_train[:round(len(X_train)*0.8)]
Y_train = Y_train[:len(X_train)]

num_samples = len(X_train)
if num_samples % batch_size != 0:
    num_samples = (num_samples // batch_size) * batch_size
    X_train = X_train[:num_samples]
    Y_train = Y_train[:num_samples]
print(len(X_train))

num_samples = len(X_val)
if num_samples % batch_size != 0:
    num_samples = (num_samples // batch_size) * batch_size
    X_val = X_val[:num_samples]
    Y_val = Y_val[:num_samples]
print("X:", len(X_val), "- Y:", len(Y_val))

history = model.fit(X_train, Y_train, batch_size=batch_size, epochs=epochs, shuffle
    =False, validation_data=(X_val, Y_val))
print(model.summary())

from sklearn.metrics import (precision_score, recall_score, f1_score,
    accuracy_score, mean_squared_error, mean_absolute_error)
from sklearn.metrics import confusion_matrix
from sklearn.metrics import multilabel_confusion_matrix

num_samples = len(X_test)
print(num_samples)
if num_samples % batch_size != 0:
    num_samples = (num_samples // batch_size) * batch_size
    X_test = X_test[:num_samples]
    Y_test = Y_test[:num_samples]

results = model.evaluate(X_test, Y_test, batch_size=batch_size)
print(results)
print("Accuracy: %.2f%%" % (results[1]*100))

model.save('LSTM_binary.keras')

plt.figure(1)
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Predicted loss and real value loss LSTM binary classification')
plt.xlabel('Epoch')
plt.ylabel('Loss persentage')
plt.legend(['Predict_loss','Real_loss'])
plt.show()
plt.savefig("loss_plt_binary.png")

plt.figure(2)
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('prediction and real accuracy LSTM binary classification')
plt.xlabel('Epoch')
plt.ylabel('Accuracy persentage')
plt.legend(['Predicted_Accuracy', 'Real_Accuracy'])
plt.show()
plt.savefig("Acc_plt_binary.png")
```

### A.2.5 Evaluate multiclass model

```python

import sys
import numpy as np
import matplotlib.pyplot as plt
from keras.models import Sequential
from keras.layers import Dense, LSTM
import tensorflow as tf
import pandas as pd
from sklearn import preprocessing
from sklearn.preprocessing import LabelEncoder
from keras.utils import to_categorical
from sklearn.metrics import (precision_score, recall_score, f1_score,
    accuracy_score, mean_squared_error, mean_absolute_error)
from sklearn.metrics import confusion_matrix
from sklearn.metrics import multilabel_confusion_matrix
from sklearn.metrics import roc_curve, auc


trained_model = sys.argv[1]
print("model: ", trained_model)
test_csv = sys.argv[2]
print("test-set: ", test_csv)

model = tf.keras.models.load_model(trained_model)
df_test = pd.read_csv(test_csv, index_col=0)
print(model.summary())

X_test = df_test[['frequency', 'volt', 'current']]
Y_test = df_test['label']

min_max_scaler = preprocessing.MinMaxScaler()
X_test = pd.DataFrame(min_max_scaler.fit_transform(X_test.values), columns=X_test.
    columns)

X_test = X_test.to_numpy()

X_test = X_test.reshape(X_test.shape[0], 1, X_test.shape[1])

encoder = LabelEncoder()
Y_test_fit = encoder.fit_transform(Y_test)
encoder.fit(Y_test_fit)
Y_test_enc = encoder.transform(Y_test)
Y_test = to_categorical(Y_test_enc, num_classes=7)

batch_size = 128
num_samples = len(X_test)
if num_samples % batch_size != 0:
    num_samples = (num_samples // batch_size) * batch_size
    X_test = X_test[:num_samples]
    Y_test = Y_test[:num_samples]


y_pred = model.predict(X_test, batch_size=batch_size)
print("ypred: ", y_pred.argmax(axis=1))
print("yActual: ", Y_test.argmax(axis=1))
#y_pred = np.argmax(y_pred, axis=1)
#print("y_pred: ", y_pred)
conf_mat = confusion_matrix(Y_test.argmax(axis=1), y_pred.argmax(axis=1))
print(conf_mat)
#tn, fp, fn, tp = conf_mat.ravel()

import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(1, figsize=(8, 6))
sns.heatmap(conf_mat, annot=True, cmap='Blues', fmt='g', xticklabels=['0', '1', '2'
    , '3', '4', '5', '6'], yticklabels=['0', '1', '2', '3', '4', '5', '6'])
plt.xlabel('Predicted labels')
plt.ylabel('True labels')
plt.title('Confusion Matrix')
plt.show()
plt.savefig("cm_plot.png")

from sklearn.metrics import classification_report
from sklearn.metrics import precision_recall_fscore_support

target_names=['class 0', 'class 1', 'class 2', 'class 3', 'class 4', 'class 5', '
    class 6']
print(classification_report(Y_test.argmax(axis=1), y_pred.argmax(axis=1),
    target_names=target_names))
#specificity = recall_score(Y_test.argmax(axis=1), y_pred.argmax(axis=1), pos_label
    =0, average='weighted')

labels=['0','1','2','3','4','5','6']
res=[]
for l in labels:
  prec,recall,_,_ = precision_recall_fscore_support(np.array(Y_test.argmax(axis=1))
    ==l, np.array(y_pred.argmax(axis=1))==l, pos_label=True,average=None)
  res.append([l, recall[0]])

pd.DataFrame(res, columns=['class','specificity'])
print(res)

plt.figure(2)
plt.plot(y_pred[0:28000:10].argmax(axis=1))
plt.plot(Y_test[0:28000:10].argmax(axis=1))
plt.title('Predictions made over time')
plt.xlabel('samples')
plt.ylabel('labels')
plt.legend(['Predicted label', 'True label'])
plt.show()
plt.savefig("predTime.png")

yPred = y_pred.argmax(axis=1)
yTrue = Y_test.argmax(axis=1)

j = 0
wrng_1 = 0
for i in yPred[0:500]:
  if i != yTrue[j]:
    wrng_1 += 1
  j += 1
wrng_2 = 0
for i in yPred[500:1000]:
  if i != yTrue[j]:
    wrng_2 += 1
  j += 1

print("1/2: ", wrng_1, "- 2/2: ", wrng_2)
```

### A.2.6 Evaluate binary model

```python

import sys
import numpy as np
import matplotlib.pyplot as plt
from keras.models import Sequential
from keras.layers import Dense, LSTM
import tensorflow as tf
import pandas as pd
from sklearn import preprocessing
from sklearn.preprocessing import LabelEncoder
from keras.utils import to_categorical
from sklearn.metrics import (precision_score, recall_score, f1_score,
    accuracy_score, mean_squared_error, mean_absolute_error)
from sklearn.metrics import confusion_matrix
from sklearn.metrics import multilabel_confusion_matrix
from sklearn.metrics import roc_curve, auc


trained_model = sys.argv[1]
print("model: ", trained_model)
test_csv = sys.argv[2]
print("test-set: ", test_csv)

model = tf.keras.models.load_model(trained_model)
df_test = pd.read_csv(test_csv, index_col=0)
print(model.summary())

X_test = df_test[['frequency', 'volt', 'current']]
Y_test = df_test['is_attack']
print(df_test['label'][0])

min_max_scaler = preprocessing.MinMaxScaler()
X_test = pd.DataFrame(min_max_scaler.fit_transform(X_test.values), columns=X_test.
    columns)

X_test = X_test.to_numpy()

X_test = X_test.reshape(X_test.shape[0], 1, X_test.shape[1])

encoder = LabelEncoder()
Y_test_fit = encoder.fit_transform(Y_test)
encoder.fit(Y_test_fit)
Y_test_enc = encoder.transform(Y_test)
Y_test = to_categorical(Y_test_enc, num_classes=2)

batch_size = 128
num_samples = len(X_test)
if num_samples % batch_size != 0:
    num_samples = (num_samples // batch_size) * batch_size
    X_test = X_test[:num_samples]
    Y_test = Y_test[:num_samples]

y_pred = model.predict(X_test, batch_size=batch_size)
print("ypred: ", y_pred.argmax(axis=1))
print("yActual: ", Y_test.argmax(axis=1))
conf_mat = confusion_matrix(Y_test.argmax(axis=1), y_pred.argmax(axis=1))
print(conf_mat)

import matplotlib.pyplot as plt
import seaborn as sns

fn_labels = [0, 0, 0, 0, 0, 0, 0]
j = 0
yPred = y_pred.argmax(axis=1)
yTest = Y_test.argmax(axis=1)

for i in yPred:
  if i != yTest[j]:
    fn_labels[int(df_test['label'][j])] += 1
  j += 1

```

```
print(fn_labels)
plt.figure(1, figsize=(8, 6))
sns.heatmap(conf_mat, annot=True, cmap='Blues', fmt='g', xticklabels=['0', '1'],
    yticklabels=['0', '1'])
plt.xlabel('Predicted label')
plt.ylabel('True label')
plt.title('Confusion Matrix')
plt.show()
plt.savefig("cm_plot_bin.png")

from sklearn.metrics import classification_report
from sklearn.metrics import precision_recall_fscore_support

target_names=['benign', 'malicious']
print(classification_report(Y_test.argmax(axis=1), y_pred.argmax(axis=1),
    target_names=target_names))
#specificity = recall_score(Y_test.argmax(axis=1), y_pred.argmax(axis=1), pos_label
    =0, average='weighted')

labels=['0','1']
res=[]
for l in labels:
  prec,recall,_,_ = precision_recall_fscore_support(np.array(Y_test.argmax(axis=1))
    ==l, np.array(y_pred.argmax(axis=1))==l, pos_label=True,average=None)
  res.append([l, recall[0]])

pd.DataFrame(res, columns=['class','specificity'])
print(res)

plt.figure(2)
plt.plot(y_pred[0:56000:10].argmax(axis=1))
plt.plot(Y_test[0:56000:10].argmax(axis=1))
plt.title('Predictions made over time')
plt.xlabel('samples')
plt.ylabel('labels')
plt.legend(['Predicted label', 'True label'])
plt.show()
plt.savefig("predTime_bin.png")
```

### A.2.7 Train and evaluate SVM model

```python
import sys
from sklearn import svm
import matplotlib.pyplot as plt
from sklearn.preprocessing import scale
import pandas as pd
import numpy as np
from sklearn import preprocessing
from joblib import dump, load
from sklearn.preprocessing import LabelEncoder
from keras.utils import to_categorical
from sklearn.metrics import (precision_score, recall_score, f1_score,
    accuracy_score, mean_squared_error, mean_absolute_error)
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix


svm_file = sys.argv[1]

if svm_file == "":
  df_train = pd.read_csv("train_simulated.csv", index_col=0, nrows=90000)
  print(df_train.head())

  X_train = df_train[['frequency', 'volt', 'current']]
  Y_train = df_train['label']

  print("scaled")
  X_train = scale(X_train)

  clf = svm.SVC(random_state=42)
  clf.fit(X_train, Y_train)
  dump(clf, 'svm.model')

else:
  clf = load('svm.model')


df_test = pd.read_csv("test_simulated.csv", index_col=0, nrows=28000)
X_test = df_test[['frequency', 'volt', 'current']]
Y_test = df_test['label']
X_test = scale(X_test)

print(X_test.shape)
print(X_test)
pred = clf.predict(X_test)
Y_test = Y_test.to_numpy()

conf_mat = confusion_matrix(Y_test, pred)
print(conf_mat)

plt.figure(figsize=(8, 6))
sns.heatmap(conf_mat, annot=True, cmap='Blues', fmt='g', xticklabels=['0', '1', '2'
    , '3', '4', '5', '6'], yticklabels=['0', '1', '2', '3', '4', '5', '6'])
plt.xlabel('Predicted labels')
plt.ylabel('True labels')
plt.title('Confusion Matrix SVM')
plt.show()
plt.savefig("cm_svm.png")

from sklearn.metrics import classification_report

target_names=['class 0', 'class 1', 'class 2', 'class 3', 'class 4', 'class 5', '
    class 6']
print(classification_report(Y_test, pred, target_names=target_names))
```

### A.2.8 Train and evaluate KNN model

```python
import sys
from sklearn import svm
import matplotlib.pyplot as plt
from sklearn.preprocessing import scale
import pandas as pd
import numpy as np
from sklearn import preprocessing
from joblib import dump, load
from sklearn.preprocessing import LabelEncoder
from keras.utils import to_categorical
from sklearn.metrics import (precision_score, recall_score, f1_score,
    accuracy_score, mean_squared_error, mean_absolute_error)
from sklearn.preprocessing import Normalizer
from sklearn.neighbors import KNeighborsClassifier
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix


knn_file = sys.argv[1]

if knn_file == "":
  df_train = pd.read_csv("train_simulated.csv", index_col=0, nrows=90000)
  print(df_train.head())

  X_train = df_train[['frequency', 'volt', 'current']]
  Y_train = df_train['label']

  transformer = Normalizer().fit(X_train)
  X_train = transformer.transform(X_train)

  print(Y_train.to_numpy())
  knn = KNeighborsClassifier(n_neighbors=5)
  knn.fit(X_train, Y_train.to_numpy())
  dump(knn, 'knn.model')
else:
  knn = load("knn.model")

df_test = pd.read_csv("test_simulated.csv", index_col=0, nrows=28000)
X_test = df_test[['frequency', 'volt', 'current']]
Y_test = df_test['label']

transformer = Normalizer().fit(X_test)
X_test = transformer.transform(X_test)

pred = knn.predict(X_test)
print(pred)

conf_mat = confusion_matrix(Y_test, pred)
print(conf_mat)

plt.figure(figsize=(8, 6))
sns.heatmap(conf_mat, annot=True, cmap='Blues', fmt='g', xticklabels=['0', '1', '2'
    , '3', '4', '5', '6'], yticklabels=['0', '1', '2', '3', '4', '5', '6'])
plt.xlabel('Predicted labels')
plt.ylabel('True labels')
plt.title('Confusion Matrix KNN')
plt.show()
plt.savefig("cm_knn.png")

from sklearn.metrics import classification_report
from sklearn.metrics import precision_recall_fscore_support

target_names=['class 0', 'class 1', 'class 2', 'class 3', 'class 4', 'class 5', '
    class 6']
print(classification_report(Y_test, pred, target_names=target_names))
```