Intrusion Detection Systems using Machine Learning and Deep Learning Techniques

A thesis submitted for the degree of

Doctor of Philosophy (PhD)

by

Hanan Hindy

Division of Cybersecurity

School of Design and Informatics

Abertay University

April, 2021

Declaration

Candidate's declarations:

I, Hanan Hindy, hereby certify that this thesis submitted in partial fulfilment of the requirements for the award of Doctor of Philosophy (PhD), Abertay University, is wholly my own work unless otherwise referenced or acknowledged. This work has not been submitted for any other qualification at any other academic institution.

Signed: [redacted]

Date: 21st April 2021.

Supervisor's declaration:

I, Natalie Coull, hereby certify that the candidate has fulfilled the conditions of the Resolution and Regulations appropriate for the degree of Doctor of Philosophy (PhD) in Abertay University and that the candidate is qualified to submit this thesis in application for that degree.

Signed: [redacted]

Date: 23rd April 2021.

Certificate of Approval

I certify that this is a true and accurate version of the thesis approved by the examiners, and that all relevant ordinance regulations have been fulfilled.

Supervisor: [redacted].

Date: 7th September 2021.

"You can't connect the dots looking forward; you can only connect them looking backwards. So, you have to trust that the dots will somehow connect in your future."

—Steve Jobs,

1955-2011

Acknowledgements

First of all, I would like to praise and thank God for his countless blessings, and for giving me the wisdom and strength to accomplish this research.

I would like to express my deep gratitude to my supervisors; Dr Xavier Bellekens, for allowing me to undertake this research and for his unceasing guidance and invaluable support, Dr Natalie Coull for believing in me and for her unlimited support, Dr Ethan Bayne, for his beneficial advice, continuous encouragement and patience, and Dr Salma Hamdy, for her help and for motivating me all through the PhD.

I would also like to sincerely thank Dr Robert Atkinson and Dr Christos Tachtatzis for their time, valuable advice, interesting and challenging conversations that enriched my knowledge, and research over the past three years.

I would like to express my appreciation to my external examiner, Prof Gordon Morison, and my internal examiner, Dr Kean Lee Kang, for their valuable comments and suggestions, that helped me improve and strengthen my thesis.

I would like to thank my mum and dad for their unconditional love and support, for bearing with me through all tough times. I thank my brother and his family for their encouragement. Thank you to all my family for looking up to me.

My friends have always been there for me, without their support, I would not have come that far. Thank you — in alphabetical order — Donia Gamal, Ghada Hamed, Kholoud ASalam, Dr Nivin Atef, Sydney Dreves, Vicky Price, Dr Wedad Hussein, and Dr Yasmine Afify.

My Dundee family made my PhD journey not only bearable but enjoyable. A genuine thank you to: Sheila & Crawford Mackenzie, Marjorie & David Dutton, Ruth & Malcolm Farquhar, Liz Higgins, and Friends International Dundee.

Last but not least, I would like to thank all my professors, colleagues, and students who never stopped supporting me.

Abstract

The increased reliance on networked technologies has led to a digital transformation of general- and special-purpose networks that further interlace technologies and heterogeneous systems. The ever-evolving technological landscape of interconnected devices constantly expands the network attack surface, which has contributed to the number and complexity of cyber attacks in recent years. The analysis of network traffic through Intrusion Detection Systems (IDS) has become an essential element of the networking security toolset. To cope with the increased rate and complexity of cyber attacks, researchers have utilised Machine Learning (ML) and Deep Learning (DL) techniques to develop IDS to cope with new and zero-day attacks. However, the lack of large, realistic, and up-to-date datasets hinders the IDS development process.

This thesis proposes an empirical investigation of ML and DL algorithms to detect known and unknown attacks in general- and special-purpose networks. The thesis further investigates how ML and DL algorithms can learn from a limited amount of data while retaining high accuracy. To this effect, a special-purpose IoT dataset is generated and evaluated against six ML techniques. The challenges and limitations of identifying anomalies in special-purpose networks are identified and discussed.

In an attempt to reduce the need for large training datasets, this thesis investigates the utilisation of Few-Shot learning paradigm to train IDS using a limited amount of data. For this purpose, Siamese networks are used and evaluated in three scenarios. This thesis further investigates the use of autoencoders to detect zero-day attacks. The zero-day attack detection experiments highlight the problem of discriminating benign-mimicking attacks. To overcome this challenge, an additional layer of feature abstraction is proposed; to improve accuracy through the cumulative aggregation of network traffic.

The results of this research demonstrate the effectiveness of the proposed approaches for IDS development. Siamese networks demonstrate their ability to learn from limited data. The proposed autoencoder models exhibit their potential to detect zero-day attacks. Finally, the significance of flow aggregation features in discriminating benign-mimicking attacks is demonstrated.

Acronyms

AI Artificial Intelligence

ANN Artificial Neural Network

ARP Address Resolution Protocol

AUC Area Under the Curve

AUROC Area Under the Receiver Operating Characteristics

CAIDA Centre for Applied Internet Data Analysis

CI Critical Infrastructure

CIC Canadian Institute for Cybersecurity

CM Confusion Matrix

CNN Convolutional Neural Network

CPS Cyber-Physical Systems

CSV Comma-Separated Values

DCS Distributed Control Systems

DDoS Distributed Denial of Service

DL Deep Learning

DNS Domain Name System

DOM Document Object Model

DoS Denial of Service

DT Decision Tree

FN False Negative

FNR False Negative Rate

FP False Positive

FPR False Positive Rate

FSM Finite State Machine

FTP File Transfer Protocol

HIDS Host Intrusion Detection System

IACS Industrial Automation and Control Systems

ICMP Internet Control Message Protocol

IDS Intrusion Detection Systems

IoT Internet of Things

IP Internet Protocol

IPS Intrusion Prevention System

k-NN k-Nearest Neighbours

LR Logistic Regression

LSTM Long Short-Term Memory

MAC Media Access Control

ML Machine Learning

MQTT Message Queuing Telemetry Transport

MSE Mean Squared Error

NB Naïve Bayes

NIDS Network Intrusion Detection System

NLP Natural Language Processing

OS Operating System

OSI Open Systems Interconnection

PCA Principal Component Analysis

PLC Programmable Logic Controllers

R2L Remote to Local

RBF Radial Basis Function

RF Random Forest

RFE Recursive Feature Elimination

RNN Recurrent Neural Network

ROC Receiver Operating Characteristics

SCADA Supervisory Control and Data Acquisition

SQL Structured Query Language

SSH Secure Shell

SSN Social Security Number

SVM Support Vector Machine

TCP Transmission Control Protocol

TN True Negative

TNR True Negative Rate

TP True Positive

TPR True Positive Rate

U2R User to Root

UDP User Datagram Protocol

URL Uniform Resource Locator

VLAN Virtual Local Area Network

VPN Virtual Private Network

XSS Cross Site Scripting

List of Symbols

x	Input
y	Output
$\boldsymbol{X}$	Input (Feature) Vector/Matrix
$\boldsymbol{Y}$	Output Vector/Matrix
C_i	i^{th} Class
$p(C_i\|x)$	Conditional Probability
$p(x\|C_i)$	Evidence
$p(C_i)$	Prior Probability
μ	Mean
σ	Standard deviation
$\|\|x\|\|_1$	L1 Norm (Manhattan Distance)
$\|\|x\|\|_2$	L2 Norm (Euclidean Distance)
η	Learning Rate
$\boldsymbol{W}$	Weights Vector/Matrix
λ	Regularisation Parameter
b	Bias
ϕ	Encoding Function
ψ	Decoding Function
$\boldsymbol{X'}$	Reconstructed Input Vector

List of Publications

[1] H. Hindy, E. Hodo, E. Bayne, A. Seeam, R. Atkinson, and X. Bellekens, "A taxonomy of malicious traffic for intrusion detection systems," in *2018 International Conference On Cyber Situational Awareness, Data Analytics And Assessment (Cyber SA)*. IEEE, 2018, pp. 1–4. https://doi.org/10.1109/CyberSA.2018.8551386

[2] H. Hindy, D. Brosset, E. Bayne, A. Seeam, and X. Bellekens, "Improving SIEM for critical SCADA water infrastructures using machine learning," in *Computer Security*. Springer, 2018, pp. 3–19. https://doi.org/10.1007/978-3-030-12786-2_1

[3] X. Bellekens, G. Jayasekara, H. Hindy, M. Bures, D. Brosset, C. Tachtatzis, and R. Atkinson, "From cyber-security deception to manipulation and gratification through gamification," in *HCI for Cybersecurity, Privacy and Trust*, A. Moallem, Ed. Cham: Springer International Publishing, 2019, pp. 99–114. https://doi.org/10.1007/978-3-030-22351-9_7

[4] C. Urquhart, X. Bellekens, C. Tachtatzis, R. Atkinson, H. Hindy, and A. Seeam, "Cyber-security internals of a Skoda Octavia vRS: A hands on approach," *IEEE Access*, vol. 7, pp. 146 057–146 069, 2019. https://doi.org/10.1109/ACCESS.2019.2943837

[5] A. Amin, A. Eldessouki, M. T. Magdy, N. Abdeen, H. Hindy, and I. Hegazy, "Androshield: Automated android applications vulnerability detection, a hybrid static and dynamic analysis approach," *Information*, vol. 10, no. 10, p. 326, Oct 2019. https://doi.org/10.3390/info10100326

[6] M. Nogues, D. Brosset, H. Hindy, X. Bellekens, and Y. Kermarrec, "Labelled network capture generation for anomaly detection," in *Foundations and Practice of Security*, A. Benzekri, M. Barbeau, G. Gong, R. Laborde, and J. Garcia-Alfaro, Eds. Cham: Springer International Publishing, 2020, pp. 98–113. https:

//doi.org/10.1007/978-3-030-45371-8_7

[7] H. Hindy, D. Brosset, E. Bayne, A. Seeam, C. Tachtatzis, R. Atkinson, and X. Bellekens, "A taxonomy of network threats and the effect of current datasets on intrusion detection systems," *IEEE Access*, 2020. https://doi.org/10.1109/ACCESS.2020.3000179

[8] H. Hindy, C. Tachtatzis, R. Atkinson, E. Bayne, and X. Bellekens, "MQTT-IoT-IDS2020: MQTT Internet of Things intrusion detection dataset," *IEEE Dataport*, 2020. https://doi.org/10.21227/bhxy-ep04

[9] J. Talbot, P. Pikula, C. Sweetmore, S. Rowe, H. Hindy, C. Tachtatzis, R. Atkinson, and X. Bellekens, "A security perspective on unikernels," in *2020 International Conference on Cyber Security and Protection of Digital Services (Cyber Security)*, 2020, pp. 1–7. https://doi.org/10.1109/CyberSecurity49315.2020.9138883

[10] H. Hindy, R. Atkinson, C. Tachtatzis, J.-N. Colin, E. Bayne, and X. Bellekens, "Utilising deep learning techniques for effective zero-day attack detection," *Electronics, Special Issue: Advanced Cybersecurity Services Design*, vol. 9, no. 10, p. 1684, Oct 2020. https://doi.org/10.3390/electronics9101684

[11] E. Ukwandu, M. A. B. Farah, H. Hindy, D. Brosset, D. Kavallieros, R. Atkinson, C. Tachtatzis, M. Bures, I. Andonovic, and X. Bellekens, "A review of cyber-ranges and test-beds: Current and future trends," *Sensors*, vol. 20, no. 24, 2020. [Online]. Available: https://www.mdpi.com/1424-8220/20/24/7148. https://doi.org/10.3390/s20247148

[12] H. Hindy, E. Bayne, M. Bures, R. Atkinson, C. Tachtatzis, and X. Bellekens, "Machine learning based IoT intrusion detection system: An MQTT case study (MQTT-IoT-IDS2020 dataset)," in *Selected Papers from the 12th International Networking Conference*, B. Ghita and S. Shiaeles, Eds. Cham: Springer International Publishing, 2021, pp. 73–84. https://doi.org/10.1007/978-3-030-64758-2_6

[13] H. Hindy, R. Atkinson, C. Tachtatzis, E. Bayne, M. Bures, and X. Bellekens, "Utilising flow aggregation to classify benign imitating attacks." *Sensors, Special Issue: Security and Privacy in the Internet of Things (IoT)*, vol. 21, no. 5, p. 1761, 2021. https://doi.org/10.3390/s21051761

[14] M. Bures, M. Klima, V. Rechtberger, B. S. Ahmed, H. Hindy, and X. Bellekens, "Review of specific features and challenges in the current Internet of Things

systems impacting their security and reliability," in *Trends and Applications in Information Systems and Technologies*. Cham: Springer International Publishing, 2021, pp. 546–556. https://doi.org/10.1007/978-3-030-72660-7_52

[15] H. Hindy, C. Tachtatzis, R. Atkinson, E. Bayne, and X. Bellekens, "Developing a Siamese network for intrusion detection systems," in *The 1st Workshop on Machine Learning and Systems (EuroMLSys) co-located with EuroSys '21, 2021*. https://doi.org/10.1145/3437984.3458842.

[16] M. Klima, V. Rechtberger, M. Bures, X. Bellekens, H. Hindy, and B. S. Ahmed, "Quality and reliability metrics for IoT systems: A consolidated view," in *EAI Urb-IoT 2020, 5th EAI International Conference on IoT in Urban Space*, 2020, *[Accepted]*.

[17] H. Hindy, C. Tachtatzis, R. Atkinson, D. Brosset, M. Bures, I. Andonovic, C. Michie, and X. Bellekens, "Leveraging Siamese networks for One-Shot intrusion detection model," *arXiv preprint arXiv:2006.15343*, 2020, *[Under Review]*.

Table of Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

1.1 Motivation

Cybersecurity is defined as the field concerned with "the protection of networks, data, and systems in the cyberspace" [1]. It is the virtual space "resulting from the interaction of people, software and services on the Internet by means of technology devices and networks connected to it" [2]. An essential component of system and network security is achieved by Intrusion Detection Systems (IDS). IDS monitor networks or systems for malicious activity or violations, and trigger alerts when a suspicious activity is detected [3]. IDS development progressed through different stages. These stages developed side by side with the increasing dependence on devices and automation, and the significant development of Machine Learning (ML) and Deep Learning (DL) techniques [4]. DL is defined as a class of neural networks that uses multiple layers to extract higher-level features allowing the modelling of complex problems [5]. Aldweesh *et al.* in their research highlight the need for "developing advanced Intrusion Detection Systems" to cope with the evolution of networks [6].

Based on Cisco's Annual Internet Report (2018-2023) [7], it is expected that an individual will have 3.6 networked devices on average in 2023 compared to 2.4 in

2018. This will result in a total of 29.3 billion networked devices. The report further discusses that attacks have grown by 776% between 100 Gbps and 400 Gbps from 2018 to 2019 and will continue to grow over the next years. With this growth of attack surface and the complexity of new attacks, current IDS however fall short of detecting new and unknown attacks.

Following the exponential rise in the number of cyber attacks and their increased complexity [8], different ML techniques were introduced to perform cyber attacks detection and classification tasks. While detection is concerned with identifying the occurrence of an attack, once detected, classification attempts to assign a label to it based on known attack classes (i.e., categorisation) [9]. Furthermore, researchers benefited from the ML advancement to develop IDS. ML techniques prove their appropriateness to build IDS, however, most of these techniques require large datasets for training and fail to flag cyber attacks that mimic benign traffic in an attempt to bypass detection mechanisms. Moreover, succeeding to the evolution of special-purpose networks, general-purpose IDS were rendered inadequate to provide detection for these networks.

Current IDS research suffers from, but not limited to, the following:

- General-purpose network IDS do not provide the security needs for special-purpose networks due to their different requirements and setup [10].
- IDS models training is timely and requires large up-to-date datasets which are difficult to obtain [11].
- Cyber attacks emerge at an exponential rate [8], therefore, by the time IDS are retrained to include new cyber attacks, more attacks may have been introduced.

Based on the limitations of current IDS research, the development of the next generation IDS is necessary, which can provide better detection capabilities. To this end, the work presented in this thesis aims to explore the utilisation of ML and DL techniques to develop the next generation IDS.

1.2 Research Objectives

IDS development evolved from signature-base to using ML techniques as cyber attacks became more complex [12]. Furthermore, the significant advancement of ML techniques benefits all research domains, which includes cybersecurity [4].

This thesis explores the suitability of using non-conventional ML and DL techniques to build the next generation of IDS. It is important to mention that, based on the literature review and the analysis of the past decade IDS [4, 13], non-conventional techniques are ones that have not been previously used for IDS development. The goal is to build models that can train using limited size data and are capable of detecting zero-day cyber attacks which are attacks that have not been previously detected or documented. Zero-Day attacks differ from unknown attacks, the latter are ones that occurred but there are not enough samples to classify them. However, zero-day attacks are ones with no previous occurrence. Zero-Day attacks can be detected by using anomaly detection (i.e., any instance that differs from normal traffic behaviour), or instances that differs from both normal traffic and known attack classes. The thesis also discusses the different challenges that accompany the processes of building IDS for special-purpose networks (e.g. Internet of Things (IoT) networks). The main objectives of this thesis can be summarised in the following research questions:

- *RQ1: How can Machine Learning be utilised to detect anomalies and attacks in special-purpose networks (i.e., IoT and Critical Infrastructure (CI))?*
- *RQ2: In an attempt to reduce the burden of needing to generate/collect large volumes of data, can IDS models train using limited-sized datasets?*
- *RQ3: In order to reduce the interim period between identifying a new cyber attack and detecting it, is there potential to build IDS that can detect new cyber attacks without retraining?*
- *RQ4: How can non-conventional DL techniques provide improved robustness and accuracy for IDS when detecting zero-day attacks?*

1.3 Thesis Statement

Building IDS is an open research field. Researchers have utilised different ML and DL techniques to build IDS, requiring large amounts of data and lengthy training processes. However, the available IDS datasets are limited and do not cover up-to-date cyber attacks. This thesis investigates the development, analysis, and evaluation of novel techniques to build IDS models that are capable of training using limit data to classify known and unknown (zero-day) attacks.

1.4 Thesis Contributions

The work presented in this thesis builds on the existing IDS research and leads towards building the next-generation IDS. The main thesis contributions are:

- A comprehensive analysis of the past decade IDS related articles. The analysis covers the most predominant IDS datasets used in the literature, the ML algorithms used for developing IDS, and cyber attacks that are covered/detected. The analysis pinpoints the shortcoming of current IDS and highlights the research gaps. To further analyse the cyber attacks coverage in IDS, an extendable cyber threat taxonomy is presented. The analysis of past decade IDS and the network threat taxonomy have been published in [13, 14].

- A model for detecting anomalies in CI networks. Six ML techniques are used, and their performances are evaluated. The models are evaluated using a real-life dataset that is collected from a water system controlled by Supervisory Control and Data Acquisition (SCADA). The experiments have been published in [15].

- The creation of an IoT IDS dataset (MQTT-IoT-IDS2020) to contribute to filling the current gap in IoT dataset availability. The dataset comprises benign traffic behaviour, generic cyber attacks, and Message Queuing Telemetry Transport

(MQTT)-based attacks. Three levels of features are extracted from the raw PCAP files; namely, packet, unidirectional flow, and bidirectional flow features. The impact of using the different feature levels on detecting generic and MQTT-based attacks is evaluated using six ML techniques. The MQTT-IoT-IDS2020 dataset is publicly available at [16] and the experiments and results have been published in [17].

- A novel utilisation of Siamese networks to build IDS. This involves the development, analysis, and evaluation of an IDS that is capable of learning from limited-size data to classify cyber attacks. Three usage scenarios are considered and evaluated. The first scenario aims to classify cyber attacks using a limited number of instances for training. The second scenario aims to classify new cyber attacks without retraining based on a few labelled instances of the new cyber attack, benefiting from One-Shot learning paradigm. Finally, the third scenario leverages similarity-based learning to detect unknown zero-day attacks. The Siamese network classification experiments have been published in [18], while the One-Shot experiments in [19].

- A model to detect zero-day cyber attacks effectively. The model relies on the encoding-decoding capabilities of autoencoders. The detection accuracy is compared with the well-established novelty detector; One-Class Support Vector Machine (SVM). These experiments have been published in [20].

- A new high level of feature abstraction, called Flow aggregation. Flow aggregation aims to benefit from the collated statistical information of individual flows. This additional feature level enhances the detection of benign-mimicking attacks, which are harder to detect because they are developed in a way that bypasses detection models. The analysis of the proposed features and their evaluation have been published in [21].

1.5 Thesis Organisation

This thesis consists of eight chapters and five appendices, which are organised as follows;

Chapter 2 presents a detailed overview of IDS. The chapter defines the core concepts that are required for the understanding of the field and this thesis. This chapter also summarises key IDS elements and attributes in a conceptual map. The conceptual map covers several aspects including IDS types, decision making, evaluation metrics. Finally, a discussion of IDS benchmark datasets is presented, which spans from the earliest KDD dataset family to the latest CICIDS dataset family.

Chapter 3 provides an analysis of recent IDS in the literature. This analysis focuses on studying the datasets of choice and the ML techniques that researchers use to build IDS models. Furthermore, the analysis presents the cyber attacks that are detected in the analysed IDS. The relation between the datasets of choice and the attack coverage is discussed. A generic cyber threat taxonomy is outlined in this Chapter. The taxonomy highlights the limitations of publicly available datasets, hindering the advancement of IDS.

Chapter 4 explores the different challenges that accompany the process of building special-purpose networks IDS *(RQ1)*. In this Chapter, six ML techniques are used to build IDS models for special-purpose networks. To this end, this chapter covers two case studies, SCADA and IoT networks. Firstly, a SCADA dataset is introduced, and three experiments are evaluated. The experiments vary based on the level of anomaly detection (i.e., binary versus multi-class detection). Secondly, a MQTT based dataset is generated and presented. The six ML techniques are used to assess the detection of generic cyber attacks and MQTT-based attacks using MQTT-IoT-IDS2020 dataset.

Chapter 5 proposes a novel One-Shot learning IDS model. Considering the problem of dataset availability and the exponential pace at which cyber attacks are introduced.

The work in this Chapter aims to leverage similarity-based learning to build IDS that can learn from limited-size data *(RQ2)*. Siamese networks are utilised, as one of the well-known One-Shot learning models, to learn pair similarity. This learning paradigm is used to not only classify cyber attacks but detect new cyber attacks using a few labelled instances without retraining *(RQ3)*. Finally, this novel model is used to detect zero-day attacks.

Following on the zero-day attack detection, Chapter 6 aims to build a zero-day detection model with high detection rate and low false positive and false negative rates *(RQ4)*. This Chapter proposes the utilisation of the encoding-decoding capabilities of autoencoders to detect zero-day attacks. The proposed model performance is compared with the well-established novelty detector model; One-Class SVM. One-Class SVM is known to perform well as an outlier, or novelty, detector, specifically with imbalanced dataset. Furthermore, Fernández *et al.* [22] demonstrate that one-class classification is effective when the minority class lacks structure, which applies to the ever-evolving zero-day attacks. As a result, One-Class SVM is expected to outperform other novelty detection methods. Therefore, only this algorithm is used as a benchmark for the algorithm developed in this chapter.

Chapter 7 addresses the problem of detecting cyber attacks that mimic benign behaviour. Benign-imitating attacks are built in a way that bypasses detection mechanisms. This Chapter proposes a higher level of feature abstraction that can assist in detecting these types of attacks.

Chapter 8 concludes the thesis by referring to the research questions in relation to all proposed models and results. Then, future work and directions are discussed.

Figure 1.1 presents the outline of the thesis chapters. It shows the dependencies and progression from one chapter to another.

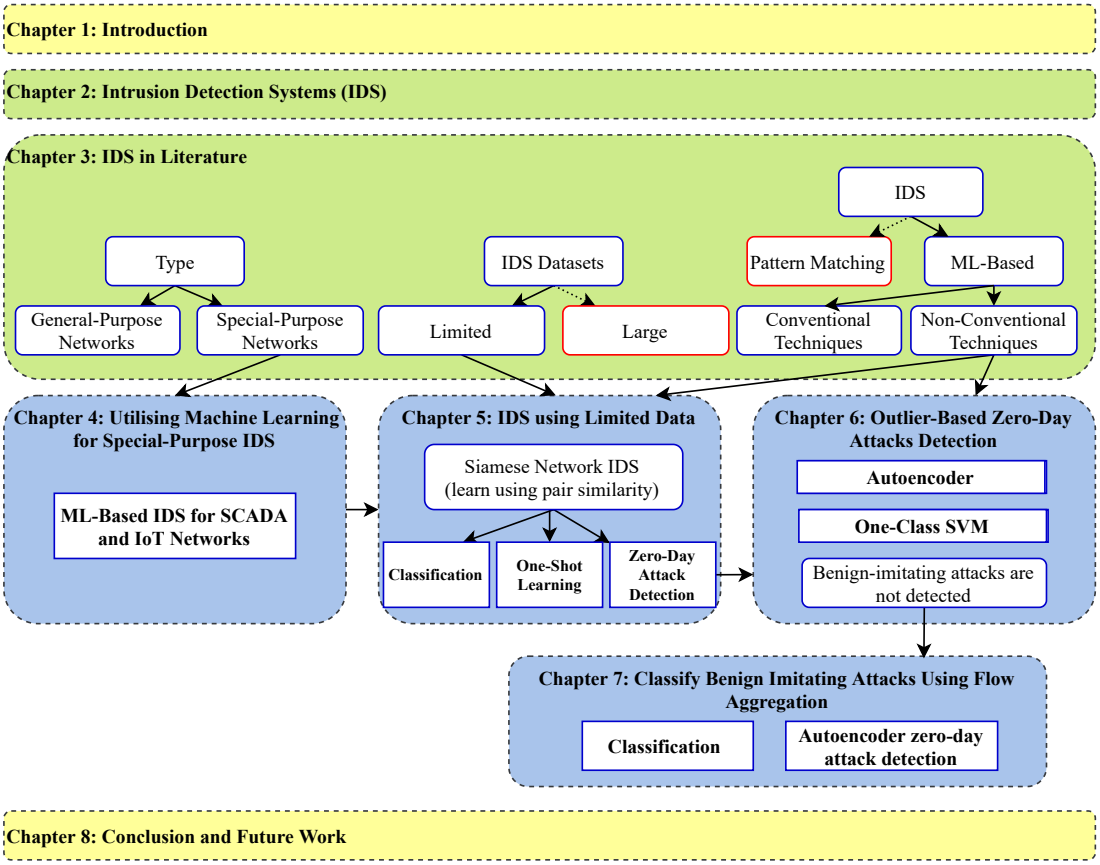


Figure 1.1
Thesis Chapters Overview
The yellow rectangles represent the introduction and conclusion, the green ones introduce the work, whereas the blue rectangles represent the work carried out within this thesis. Finally, the red bubbles represent associated fields which are not covered within the thesis.

Chapter 2

Intrusion Detection Systems

This chapter provides an explanation of the different aspects of IDS. An overview of IDS technology is presented followed by a conceptual map. The conceptual map covers the main characteristics of IDS, requirements, types, and evaluation metrics. Furthermore this section discuss different benchmark datasets, highlighting the limitations of currently available IDS datasets.

2.1 IDS Overview

IDS are systems built to monitor and analyse network traffic and/or other systems. The goal of IDS is the detection of anomalies, intrusions, or privacy violations. Ferrag *et al.* [23] represent them as the second line of defence after access control, authentication, and encryption mechanisms. IDS can either be Host Intrusion Detection System (HIDS) or Network Intrusion Detection System (NIDS). Figure 2.1 shows the two types as they differ in their monitoring scope. NIDS monitor the communication between different nodes in a network or sub-networks. They analyse the traffic flow and inwards and outwards communication. A traffic flow [24] is defined by the packets involved in the communication between two nodes in a network. A network flow could be 2-tuple, where the source and destination Internet Protocol (IP) addresses

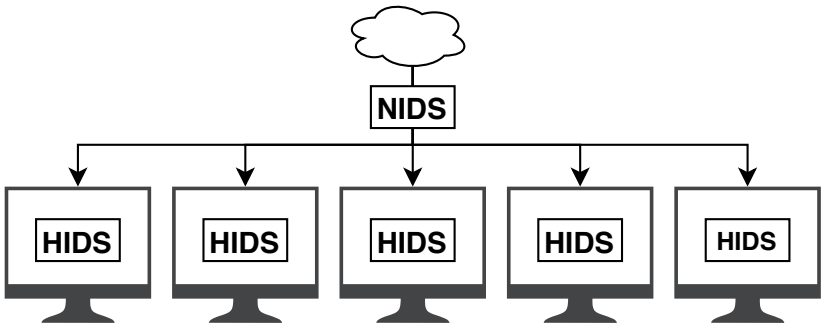


Figure 2.1
IDS Types

are used. When the source and destination ports are also used, a flow is considered to be 4-tuple, then 5-tuple flows additionally include the protocol used. Traffic flows can be unidirectional or bidirectional. Unlike NIDS, HIDS monitor node or system internals focusing on Operating System (OS) files, log files, etc., Furthermore, they can monitor the network communication of the node(s) they are installed on, which allows the analysis of encrypted traffic [25]. HIDS rely on packets content, rather than headers and/or payload information.

IDS are categorised into signature-based and anomaly-based. Signature-based IDS, also known as "Misuse Detection" [26], rely on predefined signatures that represent known intrusions and attacks. Therefore, signature-based IDS are capable of detecting attacks by comparing against known signatures. However, their detection capability is limited by the signatures available in the database used, therefore, attacks with no signature patterns go undetected; including unknown (zero-day) attacks [27].

On the other hand, anomaly-based IDS, also known as "Behaviour-based Detection" [28], depend on identifying patterns. This method requires training the system prior to deploying it. Artificial Intelligence (AI) techniques, specifically ML and DL, are well-suited for anomaly-based IDS, due to their significant training capabilities. The advantage of anomaly-based IDS is their ability to classify both normal and abnormal traffic, thus detecting known and unknown attacks. The accuracy of anomaly-based IDS against unknown attacks is better when compared

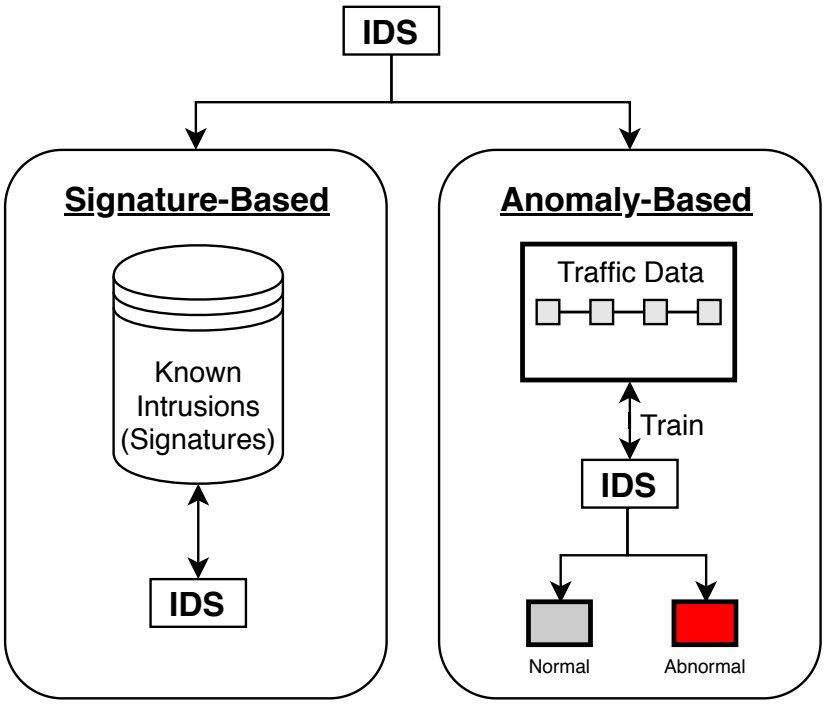


Figure 2.2
Signature-based versus Anomaly-based IDS

to signature-based IDS. However, the False Positive Rate (FPR) is often high [28]. Specification-based IDS combine the strength of both signature and anomaly-based to form a hybrid model, which can attempt to detect both known and unknown attacks using different AI techniques. Figure 2.2 compares signature-based to anomaly-based IDS. Both signature-based and anomaly-based IDS, can run on either a stateless or a stateful basis. Stateless IDS rely on packets while stateful ones rely on network flows. Recent IDS are stateful as they benefit from the "context" flows provide.

It is important to note that IDS are responsible for detecting intrusions, unlike Intrusion Prevention System (IPS) that can additionally take corrective and preventive actions [29].

In the late 80's, researchers started using statistical techniques that rely on predefined rates, as well as normal traffic that acts as a baseline for their detection. Following the use of statistical techniques, knowledge-based techniques were used, including expert systems and Finite State Machine (FSM). Finally, ML techniques

dominated the research and development of IDS. Recent surveys emphasise the focus on utilising ML and DL techniques to build IDS, including the work in [23, 30, 31, 32, 33]. The following section presents an overview of the ML pipeline prior to discussing IDS attributes and benchmark datasets.

2.2 Machine Learning Overview

ML techniques have the ability to learn patterns and behaviours and generalise decisions using a given dataset based on learning and tuning their parameters (i.e., without the need to pre-define patterns and rules). To build IDS using ML, similar to other ML applications, a multistage process is followed. This process involves preparing the data, choosing the ML model, training, validation, and testing of the chosen model. Figure 2.3 visualises the ML pipeline.

I. **Dataset Collection:** Datasets are considered the backbone of developing ML models. Large datasets are collected or generated to be used during the training and testing processes. A dataset contains raw data that can be in any format (i.e., text, audio, video, etc.). For IDS, data can be system and network log files,

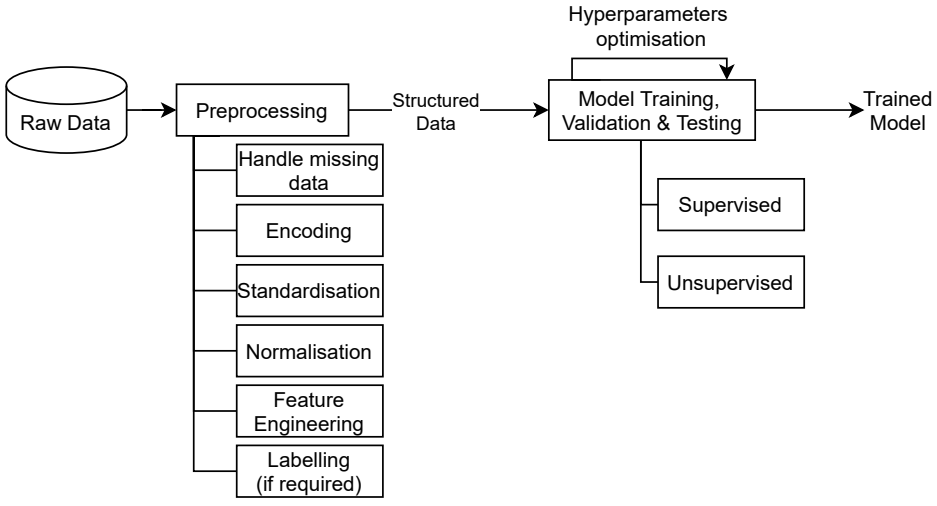


Figure 2.3
ML Pipeline

system data, and operational behaviour [34] or raw network traffic [23]. The data is structured as records (instances) and fields (features). Once a dataset is available, its instances are preprocessed.

II. **Preprocessing:** Preprocessing deals with raw data and it involves various steps to ensure that the dataset is ready for ML usage. The data can be categorised as numerical and categorical [35]. Numerical data represents quantitative values that can be either discrete (countable) or continuous (uncountable). Categorical data represents names or labels (i.e., the data that is expressed using natural language descriptions, rather than numbers). Categorical data can either be nominal or ordinal. Ordinal data values, unlike nominal one, follow a certain ranking or scale. These different data types impact decisions during the preprocessing steps.

    Preprocessing steps include:

    - *Handling missing data*: Datasets usually have missing fields/features in some instances. Instances with missing features can be dropped if they comprise a small percentage of the dataset. If a feature is missing from most of the instances, this feature can be dropped. Alternatively, several techniques can be used to fill the missing features with values, including zero or random values. In this case, random values are sampled from the same distribution, if the feature is numerical and follows a certain distribution, otherwise, values are randomly sampled within the range of given values of the rest of the instances. Statistical mean or median can be used with numerical continuous variables. The most frequent value from other instances can be used to populate missing values when it is contained in the majority of the instances. Handling missing data depends on the dataset domain, purpose, and importance of different instances and features [36, 37].

- *Encoding*: Dataset containing categorical features have to be encoded to be suitable for ML usage. The two popular encoding techniques are Ordinal and One-Hot encoding [38]. The first is used when values have ordinal relationship, otherwise, One-Hot encoding is used.
- *Normalisation*: When the distribution of a dataset feature is unknown or does not follow a Gaussian distribution, it is better to normalise/scale the values with a minimum of 0 and a maximum of 1. Normalisation, also known as Min-Max scaling [39], aims to map all values to a common scale, without distorting differences in the ranges of values. This process speeds the overall training process [40].
- *Standardisation (z-score)*: This step ensures that the feature values have a mean of 0 and a standard deviation of 1 by computing the z-score [39]. Similar to normalisation, this step ensures that all data belong on the same scale. However, outliers are not affected by standardisation.
- *Feature Engineering*: Feature Learning [41] or Feature Engineering [42] plays a vital role in building ML model since the chosen features highly affect the model performance. Contraction, extraction, and selection are the three processes that can be used to obtain features.
- *Labelling*: When a dataset is collected or generated, domain experts label the dataset instances. This step can be dropped when using unsupervised learning as labels are not used for model training.

III. **Model Training, Validation, and Testing:** ML models can be Supervised, Unsupervised, Semi-Supervised or Reinforcement learning. In supervised learning, the dataset instances are labelled (i.e., a class for each instance is known) where the model learns a function that maps input to output based on example [31]. If the output is numerical, then it is a regression model, otherwise, it is classification when the output is categorical. Classification can

either be binary (two classes), multi-class, or multi-label (class1 and class2 / class1 or class2). In unsupervised learning, data is unlabelled and the model in this case aims to discover previously undetected pattern in the training data [31]. In clustering techniques, for example, these discovered patterns are used to group instances. Supervised learning techniques include SVM, k-Nearest Neighbours (k-NN), and Decision Tree (DT), while clustering, association and dimensionality reduction are popular techniques of unsupervised learning [43].

Semi-supervised learning falls between supervised and unsupervised learning. It learns from a small amount of labelled data and a large amount of unlabelled data. Finally, reinforcement learning aims to maximise the cumulative reward while learning. This paradigm is well suited for training intelligent agents based on their actions in a certain environment.

During the training process, an ML model aims to best optimise its parameters to reach the maximum performance (i.e., accuracy) and the minimum loss (i.e., error). The dataset is split into training, validation and testing sets, as shown in Figure 2.4. The last is used to evaluate the performance of a trained model which gives an unbiased indication of how well the model is generalised [45], since evaluating using the training set can be misleading. In some cases in the literature, validation and testing sets are used interchangeably which leads to confusion [46]. However, for Artificial Neural Network (ANN), the validation set is a portion of data reserved from the training set that is later used in hyper-parameter optimisation. The testing set is only used for evaluation [46].

The randomisation of splitting of a dataset into training and testing sets has its problem as some classes and/or features can be over or underrepresented. To overcome this problem, K-fold cross validation is used. The dataset, in this case,

Figure 2.4
Dataset Split Visualisation [44]

is split into k subsets, $k-1$ subsets are used during the training process and one subset is used for testing. This process is repeated k times and the average performance is calculated [47]. K-fold cross validation is also used to estimate the average generalisation error of the model [48].

The training process continues until the model reaches the desired state. The training stops when the validation loss reaches a minimum to avoid overfitting. A model overfits when it does not generalise (i.e., its performance is limited to the training instances). This can be indicated with a low training loss and a high testing loss.

Different regularisation techniques are used to avoid overfitting and ensure that ML models generalise [49]. The three common regularisation techniques are: L1, L2, and Dropout. The difference between L1 and L2 regularisation lies in the penalty that they apply to the loss function. In L2 regularisation, also known as ridge regression, a squared magnitude penalty is applied, while in L1 regularisation, also known as lasso regression, L1 norm is applied. In the case of ANN, the magnitude is calculated based on the weights, either absolute sum (in case of L1) or sum of squares (in case of L2) [50]. The third regularisation approach, which can only be applied with ANN, is adding a Dropout layer. For a fully connected layer, all connections (weights) are trainable each iteration. Dropout randomly chooses a portion of weights to be excluded/dropped from training each iteration [51].

Feature Engineering

Features are the building blocks of dataset instances as they represent properties, variables, or attributes of data. For example, features can be number of packets, flags, duration, size, etc. Feature values construct the input to any ML or DL model. Features are obtained using one of three processes: construction, extraction, and selection.

While feature construction aims at creating new features by mining existing ones and finding missing relations within features, extraction works on raw data and/or features and applies mapping functions to extract new, representable ones. Finally, the selection aims to select the most significant subset of features. This helps reduce the feature space and required computational power.

Feature selection is done using one of three approaches [52], shown in Table 2.1; filter, wrapper, and embedded. A classification of the features used in different IDS datasets is provided in [32]. Rezaei and Liu [53] categorise features that are used for building IDS into four main categories of networking features. These categories are time series, header, payload, and statistical. Ghaffarian and Shahriari [42] consider features that represent basic network information as naïve while others are rich. Naïve features only consider attributes from packets, therefore, they do not provide enough information. However, rich features represent high level information (i.e., flow-based features) which allows them to be more discriminative.

Table 2.1
Feature Selection Approaches

Approach	Description	Advantages	Disadvantages	Examples	Ref
Filter	Selects the most meaningful features regardless of the model	Fast execution and low risk of overfitting	May choose redundant variables	- Pearson's Correlation - Chi-Square	[54]
Wrapper	Combines related variables to have subsets	Considers interactions and dependencies	Overfitting risk and high execution time	- Forward Selection - Backward Elimination -Recursive Feature Elimination	[55]
Embedded	- It is integrated as part of the model - Combines wrapper and filter methods advantages	Results in an optimal subset of variables and lower risk of overfitting	Selection is classifier dependent	- Lasso and Ridge regression - Decision Tree	[56]

2.3 IDS Conceptualisation

In this section, a broad conceptual map dedicated to the design of IDS is presented, including the different elements IDS can have. The conceptual map gives a global overview of IDS.

Figure 2.5 visualises the IDS conceptual map with each branch focusing on a dimension. Figure 2.5 (Branch 1) includes the general attributes that characterise IDS; such as its role in the network, the information provided by IDS, the system requirements, and its usage. Branch 2 describes the attributes related to the decision

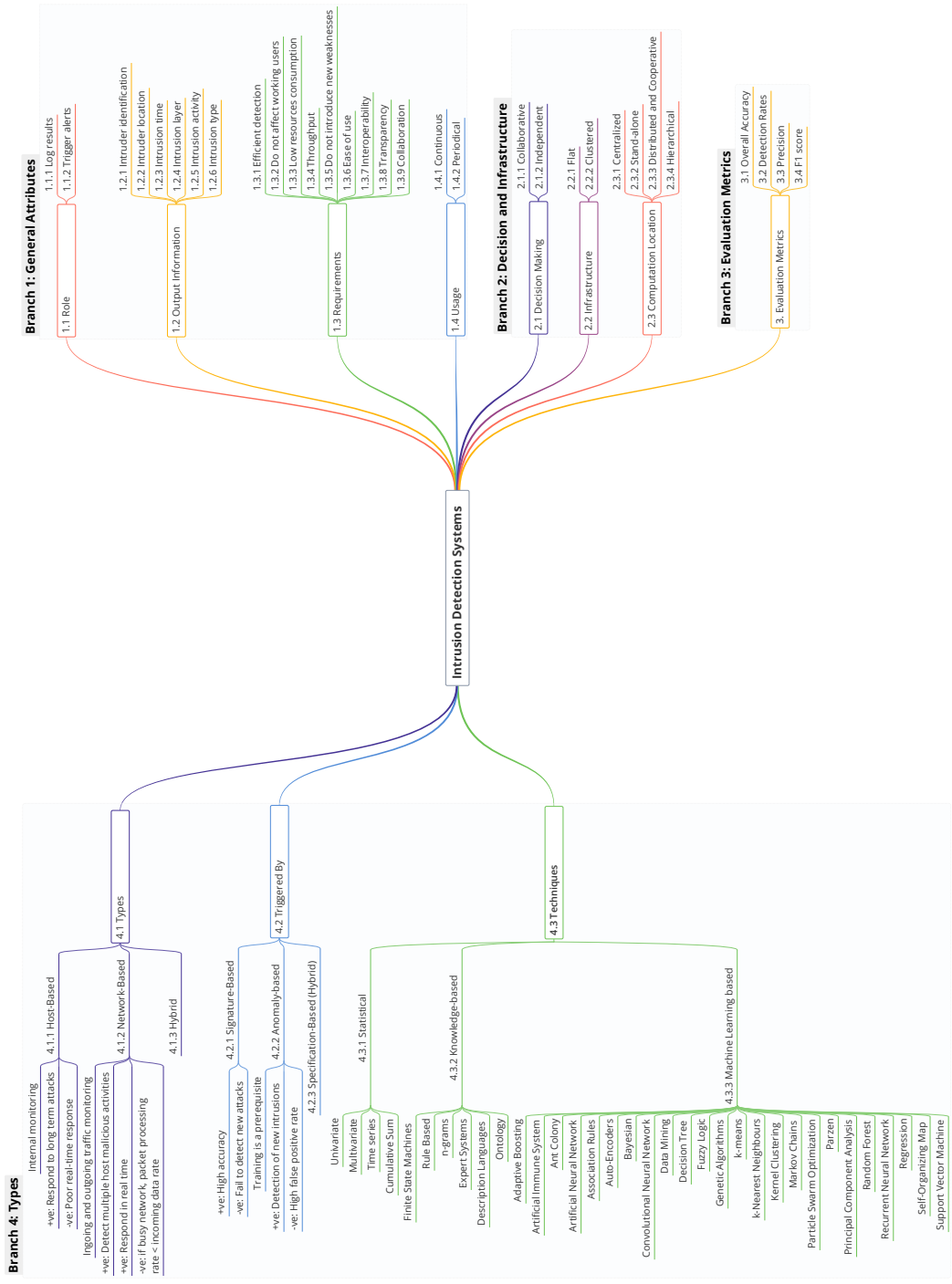


Figure 2.5
IDS Conceptual Map

types, infrastructure in place, and the computational location. Branch 3 is dedicated to IDS evaluation metrics. Finally, Branch 4 provides a descriptive analysis of IDS types including an analysis of the triggers. The different branches in Figure 2.5 are subsequently described in Sections 2.3.1 through 2.3.3.

2.3.1 General Attributes

As previously discussed, IDS focus on detecting anomalies. With reference to Figure 2.5 (Branch 1), when an intrusion is detected, IDS are expected to log the information related to the intrusion (1.1.1). These logs can then be used by network forensic investigators to further analyse the detected anomaly or enhance the learning process of IDS themselves. IDS are expected to trigger alerts upon detecting a threat (1.1.2). The alert should provide information on the detected threat and the affected system. By raising an alert, authorised users can take corrective actions and mitigate the attack.

In order to build efficient IDS, the output information provided by IDS to the end-user is critical for analysis. The recorded information should contain intruder identification information (1.2.1) and location (1.2.2) for each event. IP addresses and user credentials are used to identify the intruder. The system design should be modular to adapt to the environment. Additionally, log information can contain metadata related to the intrusion, such as timestamp (1.2.3), intrusion layer (i.e., Open Systems Interconnection (OSI)) (1.2.4), intrusion activity (1.2.5) whether the attack is active or passive and finally, the type of intrusion (1.2.6) [3]. Active attacks attempt to alter data or information in a network or system, while passive attacks monitor and gather information.

Two key aspects for effective IDS are a high detection rate (1.3.1) and a low false-positive rate. These can be evaluated using different metrics which are discussed in detail in Section 2.3.3 (Branch 3). Other important IDS factors include the

transparency (1.3.8) and safety of the overall system (1.3.2). It is crucial as an attacker may target the IDS themselves. The overall performance of IDS is also important, which includes memory requirements, power consumption (1.3.3), and throughput (1.3.4). This can highly impact IDS that are used in special-purpose networks with limited resources.

Moreover, it is crucial that IDS themselves do not introduce abnormal behaviour (1.3.5), hence a testing procedure should be set in place before deployment. The procedure can include fuzzing to detect anomalies and bugs in IDS. Such anomalies could be exploited by an attacker to render IDS useless or initiate a Denial of Service (DoS) attack [3]. Finally, Axelsson [57] adds to IDS requirements; ease of use (1.3.6), interoperability (1.3.7), transparency (1.3.8) and collaboration (1.3.9). This is important to ensure that IDS operate with other deployed security platforms.

2.3.2 Decision-Making

Figure 2.5 (Branch 2) covers the decision-making process of IDS. IDS can be distributed over multiple nodes in the network. In this case, decisions can be made collaboratively/swarm-like (2.1.1), or independently (2.1.2). In a collaborative decision-making, multiple nodes share a single decision. This collaboration can use statistical techniques such as voting and game theory, while in an independent mode, all decisions are made by individual nodes on the network [3].

Furthermore, in this distributed manner, when all nodes are working with the same capacity, it is considered a flat (2.2.1) infrastructure. Alternatively, it is a clustered infrastructure (2.2.2), where the nodes belong to clusters with different capabilities, each contributing to the decisions in a different manner. The computation location is another aspect of distributed IDS. The centralised computation location (2.3.1) works on data collected from the whole network. Unlike the centralised, the stand-alone computation (2.3.2) works on local data, disregarding decisions from other nodes.

A combination of both centralised and stand-alone can also be achieved through cooperative computation, such that each node can detect an intrusion on its own but also contributes to the overall decision [58]. Finally, IDS can also operate in hierarchical computation (2.3.4), where a cluster sends all intrusion detection information to the root node responsible for decision making [3].

2.3.3 Evaluation

A high detection rate is essential for IDS to be considered effective. However, the detection rate solely does not give a complete assessment of IDS performance.

The main elements that are used when measuring IDS performance, and hence are used to derive the different metrics, are as follows:

- True Positive (TP): Number of intrusions correctly detected
- True Negative (TN): Number of non-intrusions correctly detected
- False Positive (FP): Number of non-intrusions incorrectly detected
- False Negative (FN): Number of intrusions incorrectly detected

Hodo *et al.* [59], Buse *et al.* [26] and Aminanto *et al.* [41] discuss main IDS evaluation metrics in their respective work. These include the overall accuracy, decision rates, precision, recall, and F1-Score. IDS evaluation metrics are summarised in Figure 2.5 (Branch 3).

Overall Accuracy: Equation 2.1 provides the overall accuracy. It returns the probability that an item is correctly classified by IDS.

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

Detection Rates: Equation 2.2 calculates the Sensitivity, Specificity, Fallout, and Miss Rate detection rates, respectively. Sensitivity (TPR) calculates the probability of attack/anomaly instances that are correctly identified, while fallout (FPR) calculates the probability of incorrectly detected ones. Specificity (TPR) indicates the probability of normal/benign instances that are correctly identified, while Miss Rate (FNR) indicates the probability of incorrectly detected ones.

Detection Rates:

$$\begin{aligned} Sensitivity\ (aka\ Recall,\ True\ Positive\ Rate) &= \frac{TP}{TP+FN} \\ Specificity\ (aka\ Selectivity,\ True\ Negative\ Rate) &= \frac{TN}{TN+FP} \\ Fallout\ (aka\ False\ Positive\ Rate) &= \frac{FP}{TN+FP} \\ Miss\ Rate\ (aka\ False\ Negative\ Rate) &= \frac{FN}{TP+FN} \end{aligned} \tag{2.2}$$

Stefan Axelsson [57] emphasises the fact that high FPR (false alarm) limits the performance of IDS due to the "Base-rate fallacy problem". This problem results in neglecting alarms because the number of false positives surpasses the number of true positives.

Precision: Equation 2.3 provides the probability of positively classified incidents that are truly positive.

$$Precision = \frac{TP}{TP+FP} \tag{2.3}$$

To visualise the performance of IDS, i.e., the trade-off between sensitivity (True Positive Rate (TPR)) and fallout (True Negative Rate (TNR)), Area Under the Curve (AUC) and Receiver Operating Characteristics (ROC), also known as Area Under the Receiver Operating Characteristics (AUROC) curves are used [31, 60, 61]

F1-Score: Equation 2.4 represents the harmonic mean of precision and recall. Compared with accuracy, F1-Score does not take true negatives into account, thus it is well suited when false positive and false negative rates are critical. In addition, it is

well suited to represent the performance of IDS when dealing with imbalanced classes, such as a large number of negative instances, for example.

$$F1 = \frac{2TP}{2TP + FP + FN} \tag{2.4}$$

2.4 IDS Datasets

Researchers use benchmark datasets to evaluate IDS performance using the metrics discussed in Section 2.3.3. In this section, prominent IDS datasets are discussed. The datasets properties and limitations are highlighted.

Table 2.2 lists the prominent available IDS datasets and categorises them based on the domain they belong to. Moreover, attacks found in each are presented by tick marks. These datasets cover general-purpose IDS, Virtual Private Network (VPN), Tor Networks, Botnet, Network Flows and IoT. Details regarding the institutes contributing to the generation of these datasets and the attack types are summarised in Table A.1. The ratio between general-purpose and special-purpose IDS datasets is noticed in the table.

By observing Table 2.2, the dominance of some cyber attack classes in the datasets is clear. This is due to both their popularity and availability of tools to simulate them, which facilitates their inclusion in datasets. For example, DoS and Distributed Denial of Service (DDoS) are included in most of the datasets. The features and characteristics of these datasets are further analysed in [86]. This evaluation includes DEFCON [87], CAIDA [88], LBNL [89], CDX [90], Kyoto [91], Twente [92], UMASS [93] and ADFA [65]. Ring *et al.* [94] provide a comprehensive overview of IDS datasets, covering their main features, data format, anonymity, size, availability, recording environment, balancing, etc.

Table 2.2
IDS Prominent Datasets

General-Purpose Networks																				
									Brute-Force			Web								
Year	**Dataset**	**Normal**	**DoS**	**DDoS**	**Probe**	**U2R**	**R2L**	**Infiltrating/Scanning**	**SSH**	**FTP**	**Heartbleed**	**Brute-Force**	**XSS**	**SQL Injection**	**Webshell**	**DVWA**	**Botnet**	**Network & Host Events**	**PortScan**	**Meterpreter**
2018	CICIDS2018 [62]	✓	✓	✓	-	-	-	✓	✓	-	✓	✓	✓	✓	-	✓	✓	-	✓	-
2017	CICIDS2017 [63]	✓	✓	✓	-	-	-	✓	✓	✓	✓	✓	✓	✓	-	-	✓	-	✓	-
2017	CIC DoS dataset [64]	✓	✓	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
2017 2013	ADFA-IDS [65, 66]	✓	-	-	-	✓	✓	-	✓	✓	-	-	-	-	✓	-	-	-	-	✓
2017	Unified Network Dataset [67]	✓	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	✓	-	-
2016	DDoSTB [68]	✓	-	✓	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
2015	Booters [69]	-	-	✓	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
2015	TUIDS Coordinated Scan [70]	✓	-	-	✓	-	-	✓	-	-	-	-	-	-	-	-	-	-	-	-
	TUIDS DDoS [70]	✓	-	✓	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	TUIDS Intrusion [70]	✓	✓	-	✓	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
2014	Botnet dataset [71]	✓	-	-	-	-	-	-	-	-	-	-	-	-	-	-	✓	-	-	-
2012	STA2018 [72]	✓	✓	✓	-	-	-	✓	✓	-	-	-	-	-	-	-	-	-	-	-
2011	CTU-13 [73]	✓	-	-	-	-	-	-	-	-	-	-	-	-	-	-	✓	-	-	-
2010	ISCXIDS2012 [74]	✓	✓	✓	-	-	-	✓	✓	-	-	-	-	-	-	-	-	-	-	-
2009	Waikato [75]	-	-	✓	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
2007	CAIDA DDoS [76]	-	-	✓	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
1999	NSL-KDD [77]	✓	✓	-	✓	✓	✓	-	-	-	-	-	-	-	-	-	-	-	-	-
1999	KDD'99 [78]	✓	✓	-	✓	✓	✓	-	-	-	-	-	-	-	-	-	-	-	-	-
1998 1999 2000	DARPA [79]	✓	✓	-	✓	✓	✓	-	-	-	-	-	-	-	-	-	-	-	-	-

Special-Purpose Networks					
Year	**Dataset**	**IoT**	**VPN**	**Tor**	**SCADA**
2018	Bot-IoT [80]	✓	-	-	-
2017	Anomalies Water System [81]	-	-	-	✓
2017	IoT devices captures [82]	✓	-	-	-
2016	Tor-nonTor dataset [83]	-	-	✓	-
2016	VPN-nonVPN dataset [84]	-	✓	-	-
2015	4SICS ICS [85]	-	-	-	✓

IDS datasets can either include real (i.e., recorded from a network set-up) or synthetic (i.e., simulated or injected) traffic. Synthetic attack injection could be used to either introduce attacks to an existing dataset or balance the attack classes represented in a dataset.

The three main issues with current IDS datasets are that (i) they lack real-life characteristics of recent network traffic, which renders current IDS not applicable for production environments [95], (ii) there is a limited number of datasets for special-purpose networks (i.e. IoT and CI) which again limits the IDS development, and (iii) they do not cope with constantly changing networks topology.

Viegas *et al.* [95] mention that for a dataset to be considered appropriate, it has to possess various properties; instances should be labelled, the dataset should contain real network traffic, can be reproducible, and shareable, implying that the dataset should not contain any confidential data.

The analysis of IDS research articles in the past years is overviewed in the following chapter. The chapter discusses the datasets of choice and the dominant techniques used to build IDS. Furthermore, the effect of dataset choice on the advancement of IDS research is outlined.

2.5 Summary

In this chapter, an introduction and an overview of IDS and ML concepts are provided. IDS classification, types, and learning paradigms are highlighted. Furthermore, the discussion is elaborated and extended using the IDS conceptual map. The conceptual map covers various aspects of building IDS which involves IDS evaluation metrics, used techniques, and datasets are analysed. Finally, IDS datasets are reviewed where prominent IDS datasets are outlined. The discussion pinpoints the limitations of existing datasets; including the lack of special-purpose ones and the dominant representation of some cyber attacks like DoS. The effect of datasets and a thorough discussion of recent IDS follows in the following chapter.

Chapter 3

IDS in Literature

In this chapter, IDS research articles are discussed and analysed with respect to the different datasets used and the ML algorithms to train IDS. Following on from the analysis, an overview of the cyber attacks that are detected by recent IDS is presented. The cyber attack coverage is conducted not only with respect to the analysed articles, but also in relation to a generic cyber threat taxonomy. The presented taxonomy classifies cyber threats based on the OSI layers, active or passive behaviour and threat source. In active attacks, the attacker attempts to modify the data or impact the network or system performance, while in passive attacks, the aim is to observe and gather information for further analysis and usage. The comprehensive taxonomy is built in an extendable fashion and has been released publicly for future amendments. Finally, the chapter highlights the overall trends and limitations of IDS research in recent years, which influence the research presented in the rest of this thesis.

3.1 Analysis of Recent IDS Research

In this section, recent IDS articles are discussed, analysing both research trends and shortcomings. This analysis highlights the main algorithms used and the datasets of choice, which concludes the strengths and weaknesses of recent IDS. To this end,

IEEE Xplore and Google Scholar queries were made using "Intrusion Detection System*" OR "IDS*", further filtering results to include articles published in the range [2008-2020]. The filtration was made to have a wide coverage of datasets, ML techniques, and detected attacks. A total of **90** published articles in this period were analysed. Analysis of older IDS ML techniques and used features for the period [2004-2007] was previously conducted by Nguyen and Armitage [31]. They discuss the limitations of port-based and payload-based classification and the emerging use of ML techniques to classify IP traffic.

Table 3.1 summarises the IDS research selected based on the above criteria. Each row represents one article, highlighting the dataset(s) and algorithm(s) used within the research, alongside the attacks that IDS are capable of detecting. The algorithm trends are discussed later in this section alongside the attacks' coverage in the datasets used. It is important to note that Table 3.1 is used to provide insights regarding the analysed IDS.

Table 3.1
Over A Decade of IDS [2008 - 2020]

Year	Dataset	Used Algorithms	Detected Attacks	Ref
2008	KDD-99	- Tree Classifiers - Bayesian Clustering	Probing, DoS, R2L, U2R	[96]
2008	KDD-99	- v-SVC - K-Means - Parzen Classifier	Probing, DoS, R2L, U2R	[97]
2008	- PIERS - Emergency Department Dataset - KDD-99	- Bayesian Network Likelihood - Conditional Anomaly Detection - WSARE	APD: - Illegal activity in imported containers - Anthrax - DoS, R2L	[98]
2008	KDD-99	AdaBoost	Probing, DoS, R2L, U2R	[99]
2009	KDD-99	- ABC - Fuzzy Association Rules	Probing, DoS, R2L, U2R	[100]

Continued . . .

Year	Dataset	Used Algorithms	Detected Attacks	Ref
2009	Collected transactions dataset	Fuzzy Association Rules	Credit Card Fraud	[101]
2009	KDD-99	Genetic-based	Probing, DoS, R2L, U2R	[102]
2009	KDD-99	C4.5	Probing, DoS, R2L, U2R	[103]
2009	KDD-99	BSPNN using: - AdaBoost - Semi-parametric NN	Probing, DoS, R2L, U2R	[104]
2009	1999 DARPA	- RBF - Elman NN	Probing, DoS, R2L, U2R	[105]
2009	1999 DARPA	- SNORT - Non-Parametric CUSUM - EM based Clustering	13 Attack Types	[106]
2010	KDD-99	FC-ANN based on: - ANN - Fuzzy Clustering	Probing, DoS, R2L, U2R	[107]
2010	KDD-99	Logistic Regression	Probing, DoS, R2L, U2R	[108]
2010	KDD-99	- NN - FCM Clustering	Probing, DoS, R2L, U2R	[109]
2011	Generated dataset	OCSVM	Scan (Nachi, Netbios, SSH), TCP flood, DDoS (TCP, UDP flood), Stealthy DDoS UDP flood, Traffic deletion, Popup spam	[110]
2011	KDD-99	- NB - AdaBoost	Probing, DoS, R2L, U2R	[111]
2011	KDD-99	- Weighted k-NN - Genetic Algorithm	DoS / DDoS	[112]
2011	KDD-99	Genetic Fuzzy Systems based on: - IRL - Michigan - Pittsburgh	Probing, DoS, R2L, U2R	[113]
2011	KDD-99	- DT - RBF NN - NB - Ripper Rule - BON	Probing, DoS	[114]
2011	KDD-99	- SOM - K-Means clustering	Probing, DoS, R2L, U2R	[115]

Continued . . .

Year	Dataset	Used Algorithms	Detected Attacks	Ref
2011	KDD-99	- Rule-Based - ART Network - BON	Probing, DoS, R2L, U2R	[116]
2011	KDD-99	SVM	Probing, DoS, R2L, U2R	[117]
2011	KDD-99	- NB - K-Means	Probing, DoS, R2L, U2R	[118]
2012	KDD-99	- K-Means - Modified SOM	Probing, DoS, R2L, U2R	[119]
2012	1998 DARPA	SVM	Attack, Non-Attack	[120]
2012	1998 DARPA	ELMs: - Basic - Kernel-Based	Probing, DoS, R2L, U2R	[121]
2012	1998 DARPA	SVDD	U2R	[122]
2012	KDD-99	Hidden NB	Probing, DoS, R2L, U2R	[123]
2012	KDD-99	- DT - SVM - SA	Probing, DoS, R2L, U2R	[124]
2012	KDD-99	Ensemble DTs: - NB Tree - RF - Random Tree - C4.5 - Decision Stump - Representative Tree model	Probing, DoS, R2L, U2R	[125]
2012	KDD-99	- K-Means - SVM - Ant Colony	Probing, DoS, R2L, U2R	[126]
2013	KDD-99	- Fuzzy C means - Fuzzy NN / Neurofuzzy - RBF SVM	Probing, DoS, R2L, U2R	[127]
2013	NSL-KDD	Fuzzy Clustering NN	Probing, DoS, R2L, U2R	[128]
2013	KDD-99	- K-Means - NN MLP	Probing, DoS, R2L, U2R	[129]
2013	KDD-99	- FFNN - ENN - GRNN - PNN - RBNN	Probing, DoS, R2L, U2R	[130]
2013	DARPA 2000	APAN using: - Markov Chain - Kmeans Clustering	DDoS	[131]
2013	ISCX 2012	KMC+NBC - NB Classifier - K-Means Clustering	Normal, Attack	[132]

Continued . . .

Year	Dataset	Used Algorithms	Detected Attacks	Ref
2013	Bank's Credit Card Data	DT	Fraud	[133]
2013	KDD-99	Two variants of GMDH: - Monolithic - Ensemble-based	Probing, DoS, R2L, U2R	[134]
2013	Simulated dataset	Non-Parametric CUSUM	Jamming	[135]
2014	KDD-99	ELM	Probing, DoS, R2L, U2R	[136]
2014	- KDD-99 - NSL-KDD	ANN-Bayesian Net-GR ensemble: - ANN - Bayesian Net with GR feature selection	Probing, DoS, R2L, U2R	[137]
2014	NSL-KDD	- C4.5 DT - One-class SVM	Probing, DoS, R2L, U2R	[138]
2014	KDD-99	K-medoids	Probing, DoS, R2L, U2R	[139]
2014	KDD-99	- SVM - CSOACN	Probing, DoS, R2L, U2R	[140]
2014	NSL-KDD	AIS (NSA, CSA, INA)	Normal, Abnormal	[141]
2015	KDD-99	- DT - CFA (Feature Selection)	Probing, DoS, R2L, U2R	[142]
2015	gureKddcup6 percent	SVM	R2L	[143]
2015	KDD-99	- K-Means - k-NN	Probing, DoS, R2L, U2R	[144]
2015	KDD-99	Weighted ELM	Probing, DoS, R2L, U2R	[145]
2015	GureKddcup	AIS (R-chunk)	Normal, Abnormal	[146]
2016	KDD-99	- k-NN - PCA - Fuzzy PCA	Probing, DoS, R2L, U2R	[147]
2016	NSL-KDD	- NB - PCA - MLP - SVM - C4.5	Probing, DoS, R2L, U2R	[148]
2016	Simulated dataset	ANN	DoS/DDoS	[149]

Continued . . .

Year	Dataset	Used Algorithms	Detected Attacks	Ref
2016	Generated dataset using httperf	Static and Dynamic Mapping	SQL Injection, XSS	[150]
2016	KDD-99	- SVM - PCA	- Normal, Attack	[151]
2016	NSL-KDD	- SVM - DT (J48) - AIS (NSA-GA) - NB	Normal, Abnormal	[152]
2017	- Kyoto2006+ - NSL-KDD	- Forked VAE - Unsupervised Deep NN	Probing, DoS, R2L, U2R	[153]
2017	KDD-99	- Binary PSO - k-NN	Probing, DoS, R2L, U2R	[154]
2017	KDD-99	- R-tree - k-NN - K-Means - SVM	Probing, DoS, R2L, U2R	[155]
2017	Generated dataset	- BON - GPU-based ANN	Normal, Attack	[156]
2017	NSL-KDD	DL RNN	Probing, DoS, R2L, U2R	[157]
2017	NSL-KDD	- K-Means - NB - Information Gain	Probing, DoS, R2L, U2R	[158]
2017	UNB-CIC	- ANN - SVM	nonTor Traffic	[159]
2017	KDD-99	Polynomial Feature Correlation	DoS	[160]
2017	KDD-99	- PCA - k-NN - Softmax Regression	Probing, DoS, R2L, U2R	[161]
2017	KDD-99	Optimized Backpropagation by Conjugate Gradient algorithm - Fletcher Reeves - Polak Ribiere - Powell Beale	Probing, DoS, R2L, U2R	[162]
2017	NSL-KDD	Denoising Auto-Encoder	Normal, Anomaly	[163]
2018	KDD-99	Kernel Clustering	Probing, DoS, R2L, U2R	[164]

Continued . . .

Year	Dataset	Used Algorithms	Detected Attacks	Ref
2018	Simulated Dataset	- MLP - NB - SVM - J48 - Logistic - RF Features Selection: - BFS-CFS - GS-CFS	Individual and Combination Routing Attacks: - Hello Flood, Sinkhole, Wormhole	[165]
2018	KDD-99	- FLN - PSO	Probing, DoS, R2L, U2R	[166]
2018	- NSL-KDD - UNSW-NB15	- ANN - Deep Auto-Encoder	- Probing, DoS, R2L, U2R - Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode, Worm	[167]
2018	- KDD-99 - NSL-KDD	- DL - NDAE - Stacked NDAEs	Probing, DoS, R2L, U2R	[168]
2018	KDD-99	- SMO - RF - MFNN - NB - KFRFS - IBK - AdaBoost	Probing, DoS, R2L, U2R	[169]
2018	NSL-KDD	AIS (NSA, CSA)	Normal, Abnormal	[170]
2018	- KDD-99 - CAIDA'07/08 - Generated traffic	AIS	DoS	[171]
2019	NSL-KDD	- NB -ANN - RF - SVM - BayesNet - DT (Enhanced J48, J48, ADTree, DecisionStump, RandomTree, SimpleCart)	Probing, DoS, R2l, U2R	[172]
2019	KDD-99	- DT - SVM (least square) Feature Selection: - FGLCC - CFA	Probing, DoS, R2l, U2R	[173]

Continued . . .

Year	Dataset	Used Algorithms	Detected Attacks	Ref
2019	- UNSW-NB15 - CICIDS2017	- RF - Deep FFNN - Gradient Boosting Tree	- Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shell-Code and Worms - DoS, DDoS, Web-based, Brute-force, Infiltration, Heartbleed, Bot and Scan	[174]
2019	- ISCX 2012 - NSL-KDD - Kyoto2006+	- IG - PCA - SVM - IBK - MLP	- Normal, Attack - Probing, DoS, R2l, U2R	[175]
2019	- KDD-99 - NSL-KDD - UNSW-NB15 - Kyoto2006+ - WSN-DS - CICIDS2017	- NB - DT - RF - AB - k-NN - SVM - Deep NN - Logistic Regression	- Probing, DoS, R2l, U2R - 4 DoS attacks (Blackhole, Grayhole, Flooding and Scheduling) - Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shell-Code and Worms - DoS, DDoS, Web-based, Brute-force, Infiltration, Heartbleed, Bot and Scan	[176]
2019	NSL-KDD	- DT - k-NN - MLP - SVM - Kernel ELM - Genetic Algorithms	Probing, Dos, R2L, U2R	[177]
2019	NSL-KDD	Auto-Encoder	Normal, Anomaly	[178]
2019	- KDD-99 - NSL-KDD - Kyoto - UNSW-NB15 - WSN-DS - CICIDS2017 - ADFA-LD - ADFA-WD	- LR - NB - DT - RF - k-NN - SVM - DNN - AdaBoost	- Probing, Dos, R2L, U2R - Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shell-Code and Worms - 4 DoS attacks (Blackhole, Grayhole, Flooding and Scheduling) - DoS, DDoS, Web-based, Brute-force, Bot and Scan	[176]
2019	CICIDS2017	- SMOTE - EFS - AdaBoost - PCA	DDoS	[179]

Continued ...

Year	Dataset	Used Algorithms	Detected Attacks	Ref
2020	Generated dataset	- RF - J48 - BayesNet - SVM - AdaBoost - MLP - Decision Stump - NB	DDoS	[180]
2020	NSL-KDD	Deep NN	Probing, DoS, R2l, U2R	[181]
2020	KDD-99	- NB - DT - Ontology - RF	Probing, Dos, R2L, U2R	[182]
2020	Generated dataset	- Isolation Forest - Local Outlier Factor	Port Scanning, HTTP and SSH Brute-Force, SYN Flood	[183]
2020	CICIDS2017	- DT - NB - MLP - RF - J48 - LSTM - k-NN	SSH and FTP Brute-force , Web Attacks (Brute-force, XSS and SQL Injection)	[184]

Where:
- ABC: Association-Based Classification
- AdaBoost : Adaptive Boosting
- AIS: Artificial Immune System
- ANN: Artificial Neural Network
- APAN: Advanced Probabilistic Approach for Network-based IDS
- APD: Anomaly Pattern Detection
- ART: Adaptive Resonance Theory
- BFS-CFS: Best First Search with Correlation Features Selection
- BON: Back-Propagation Network
- BSPNN: Boosted Subspace Probabilistic NN
- CFA: CuttleFish Algorithm
- CSA: Clonal Selection Algorithm
- CSOACN: Clustering based on Self-Organized Ant Colony Network
- CUSUM: CUmulative SUM
- DL: Deep Learning
- DoS: Denial of Service
- DT: Decision Tree
- EFS: Ensemble Feature Selection
- ELM: Extreme Learning Machine
- ENN: Elman NN
- FCM: Fuzzy C-Mean
- FFNN: Feed Forward NN
- FGLCC: Feature Grouping based on Linear Correlation Coefficient
- FLN: Fast Learning Network
- GA: Genetic Algorithm
- GMDH: Group Method for Data Handling
- GR: Gain Ratio
- GRNN: Generalised Regression NN
- GS-CFS: Greedy Step-wise with Correlation Features Selection
- IG: Information Gain
- INA: Immune Network Algorithms
- IRL: Iterative Rule Learning
- KFRFS: Kernel-based Fuzzy-Rough Feature Selection
- k-NN: k-Nearest Neighbours
- MFNN: Multi-Functional Nearest-Neighbour
- MLP: Multi-Layer Perceptron
- NB: Naïve Bayes
- NDAE: Non-Symmetric Deep Auto-Encoder
- NN: Neural Network
- NSA: Negative Selection Algorithm
- OCSVM: One-Class Support Vector Machine
- PCA: Principal Component Analysis
- PNN: Probabilistic NN
- PSO: Particle Swarm Optimisation
- R2L: Remote to Local
- RBF: Radial Basis Function
- RBNN: Radial Basis NN
- RF: Random Forest
- RNN: Recurrent Neural Networks
- SA: Simulated Annealing
- SOM: Self-Organising Map
- SMO: Sequential Minimal Optimisation
- SMOTE: Synthetic Minority Oversampling Technique
- SVDD: Support Vector Data Description
- SVM: Support Vector Machine
- U2R: User to Root
- VAE: Variational Auto-Encoder
- WSARE: What's Strange About Recent Events
- XSS: Cross Site Scripting

Figure 3.1 shows the IDS datasets distribution based on the usage in the past decade, highlighting the percentage of each. Only 10% of the IDS use simulated and unpublished datasets. This results in IDS that neither cover real-life situations nor suit constantly changing networks and special-purpose networks. This signifies that the developed IDS are not deployable, and only limited for research purposes.

The figure also highlights a noticeable preference of the KDD dataset family as nearly 47% of the selected publications use the KDD Cup'99 dataset and 18% use the NSL-KDD dataset. Excluding the unpublished and simulated datasets, the second most used dataset is the DARPA. This inclination is owed to the datasets availability and popularity. Moreover, they are the oldest benchmark datasets, hence, researchers tend to use them for evaluation and comparisons [23]. Another reason for this inclination is the datasets' practicality; the KDD dataset family contains normal instances and four attacks; namely, DoS, User to Root (U2R), Remote to Local (R2L), and probing, with multiple categories of each attack and the features are already processed, extracted, and presented in a ML-ready format.

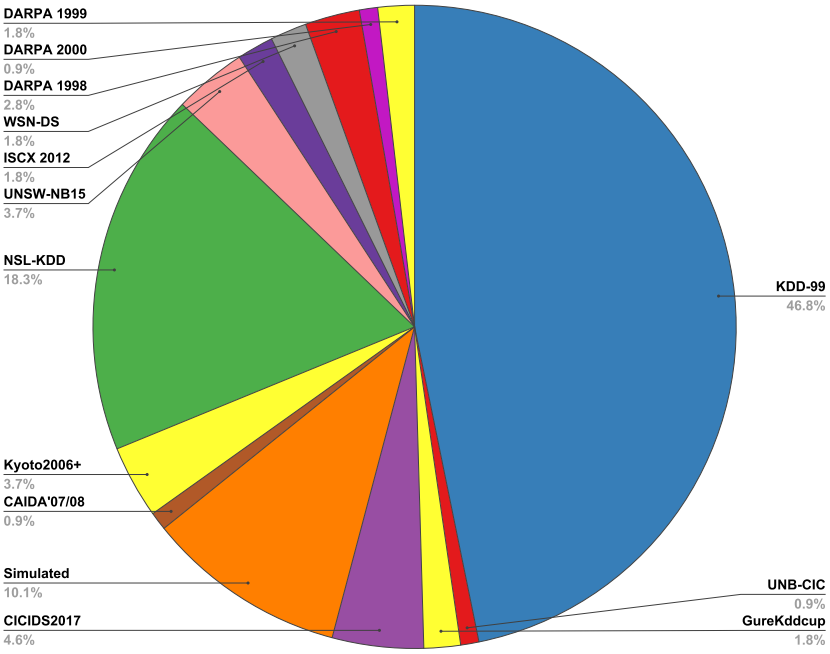


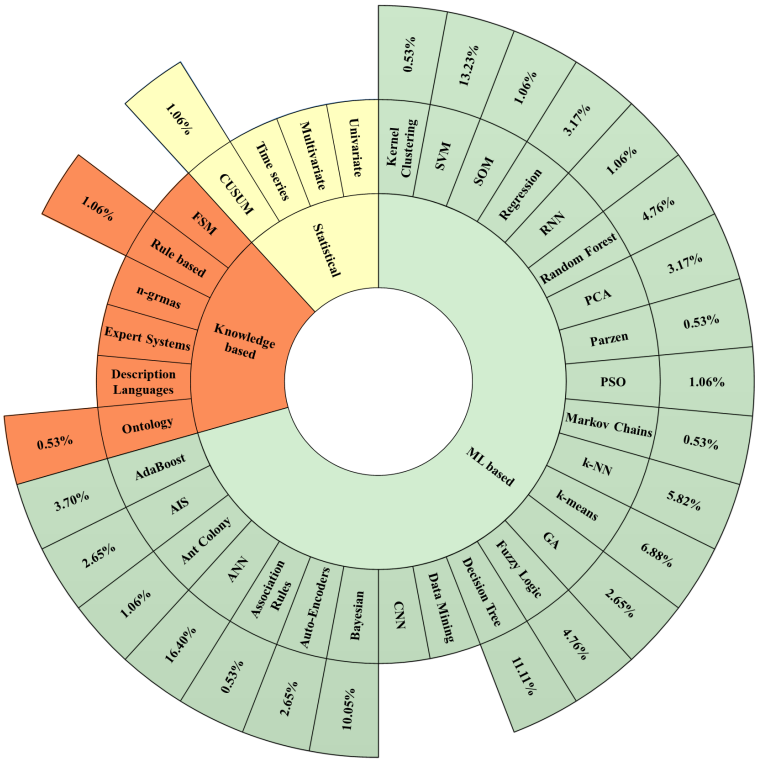
Figure 3.1
Distribution of Datasets Used for IDS Evaluation from Articles Listed in Table 3.1

Despite their popularity, researchers acknowledge these datasets' shortcomings. The datasets fail to accurately represent current attacks because they were generated in the late 90s. Moreover, their use leads to an endemic situation; numerous results reported in the literature claim detection results which are not applicable in real-world scenarios. Al Tobi and Duncan [185] provide a comprehensive analysis of the drawbacks of the KDD Cup'99 dataset. The shortcomings of the DARPA dataset are analysed by Mahoney and Chan [186] and McHugh [187]. Alongside the limitations of each dataset, they are also deprecated, hence, confirming the inability of most of the IDS presented in Table 3.1 to cope with recent attacks and threats.

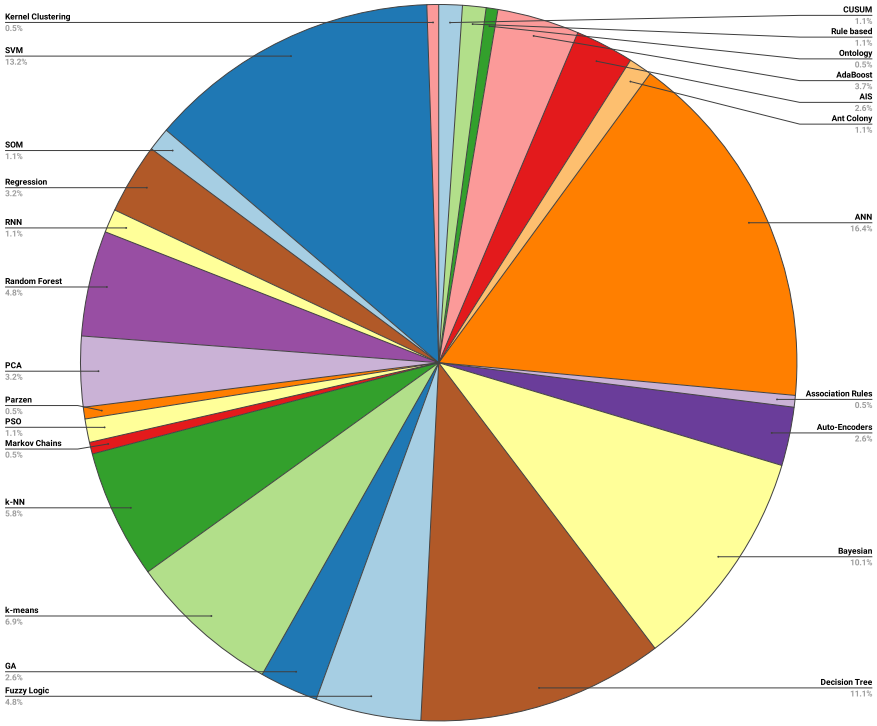
To further analyse the last decade's research on IDS, the detection algorithms in the selected articles are considered. Anomaly-based and specification-based IDS are based on identifying patterns that discriminate normal from abnormal traffic and distinguish different attack classes. These IDS can be subcategorised based on the training method used as previously visualised in Figure 2.5. The two charts in Figure 3.2 are constructed to investigate how well the current literature covers the range of technique categories collated in the conceptual map in Figure 2.5.

From the centre moving outwards, Figure 3.2a shows the three main categories of algorithms and their corresponding subcategories in line with the conceptual map. The outer circle shows the percentage of IDS from Table 3.1 that use these algorithms. The chart highlights the dominance of ML algorithms employed when building IDS. As shown, both statistical and knowledge-based algorithms are less represented. This dominance is due to (i) the sophistication of new cyber attacks which poses the need for more complex detection techniques [12] and (ii) the significant advancement of ML techniques in various research domains that involve cybersecurity [4].

Figure 3.2b on the other hand, plots the distribution of those algorithms that are actually used in the literature according to Table 3.1. The plot shows the dominance of ANN, SVM, and k-means as the most used algorithms. This is reasoned by their



(a) Occurrence of all algorithms categories based on Figure 2.5



(b) Distribution of used algorithms discussed in Table 3.1

Figure 3.2
Distribution of Algorithms Usage in the IDS from Articles Listed in Table 3.1

ability to discriminate between benign and attack classes given a feature set. However, leveraging new/emerging ML techniques and adapting ones from other domains will advance the development of the next generation IDS; in a matter of benefiting from the advancement and knowledge of ML in these domains.

3.2 Threats Taxonomy

One of the first attacks classifications was proposed by Kendall [188]. They classified intrusions into four categories, namely: DoS, R2L, U2R and Probing. This aligns with the KDD dataset family and can be noticed by observing the dataset family timeline provided by Siddique *et al.* [189].

Following this, multiple other classifications were suggested in the literature. These classifications focus on specific aspects of attacks or an explicit target domain. For example, Welch and Lathrop [190] classifies threats in wireless networks based on attack techniques, resulting in seven different categories. These are: Traffic Analysis, Passive Eavesdropping, Active Eavesdropping, Unauthorised Access, Man-in-the-middle, Session Hijacking and Replay. IoT security requirements were the motivation for the threats classification by Sachin Babar *et al.* [191]. These requirements are: identification, communication, physical threat, embedded security, and storage management.

Despite the availability of traditional threat taxonomies, the need for a recent and extendable one arose from the prevalence of common attacks found in current IDS datasets, as illustrated in Section 2.4. The absence of a modern cyber threat taxonomy additionally presents a further challenge for researchers in ascertaining the threat coverage of existing datasets. Building a generic and modular taxonomy for security threats can assist researchers and cybersecurity practitioners build tools that are more capable of identifying a more comprehensive subset of attacks, including known, advanced, and new zero-day attacks.

In this thesis, a new, extendable taxonomy is proposed to categorise network threats based on (i) source of the threat, (ii) the affected OSI model layers, and (iii) active or passive threat. The taxonomy is depicted in Figure 3.3, and although it places attacks under a single target layer of the OSI model, it is important to highlight that other layers may also be affected. The focus here is on the main target layer of attack.

An attack is interpreted to be active if it alters or changes any aspect of a network or a system. For example, it can disturb the performance or affect information. During passive attacks, the network resources are left intact and the attacker is concerned with either gathering information or monitoring the network. Active threats are shown in Figure 3.3 as *rectangles* while passive ones are represented by *ovals*. Examples of active attacks include DoS and DDoS (Figure 3.3 - 1.1) and Impersonation (Figure 3.3 - 1.5). Examples of passive attacks comprise Scanning (Figure 3.3 - 1.6) and probing (Figure 3.3 - 1.9). Some attacks cannot be identified as either active or passive until their impact is known. Code injections (Figure 3.3 - 3.1), for example, are considered passive attacks when the code is used to query data or gather information, and active if the code changes data or alters a database schema by dropping tables or relations.

The following subsections elaborate on the five threat sources included in the taxonomy, and the different attacks branching under each.

3.2.1 Network Threats

Network threats are initiated based on a flow of packets sent over a network. The most popular network threats are DoS and DDoS (Figure 3.3 - 1.1). In a DoS attack, an attacker prevents legitimate users from accessing a certain service by flooding the network with requests. As a result, the service/server under attack looks unresponsive. In DoS, a single machine is used to perform the attack, however, multiple machines are used to initiate a DDoS attack. DDoS attacks are usually confused with a common anomaly called “Flash Crowds” [192]. Flash crowds occur when a high flow of traffic

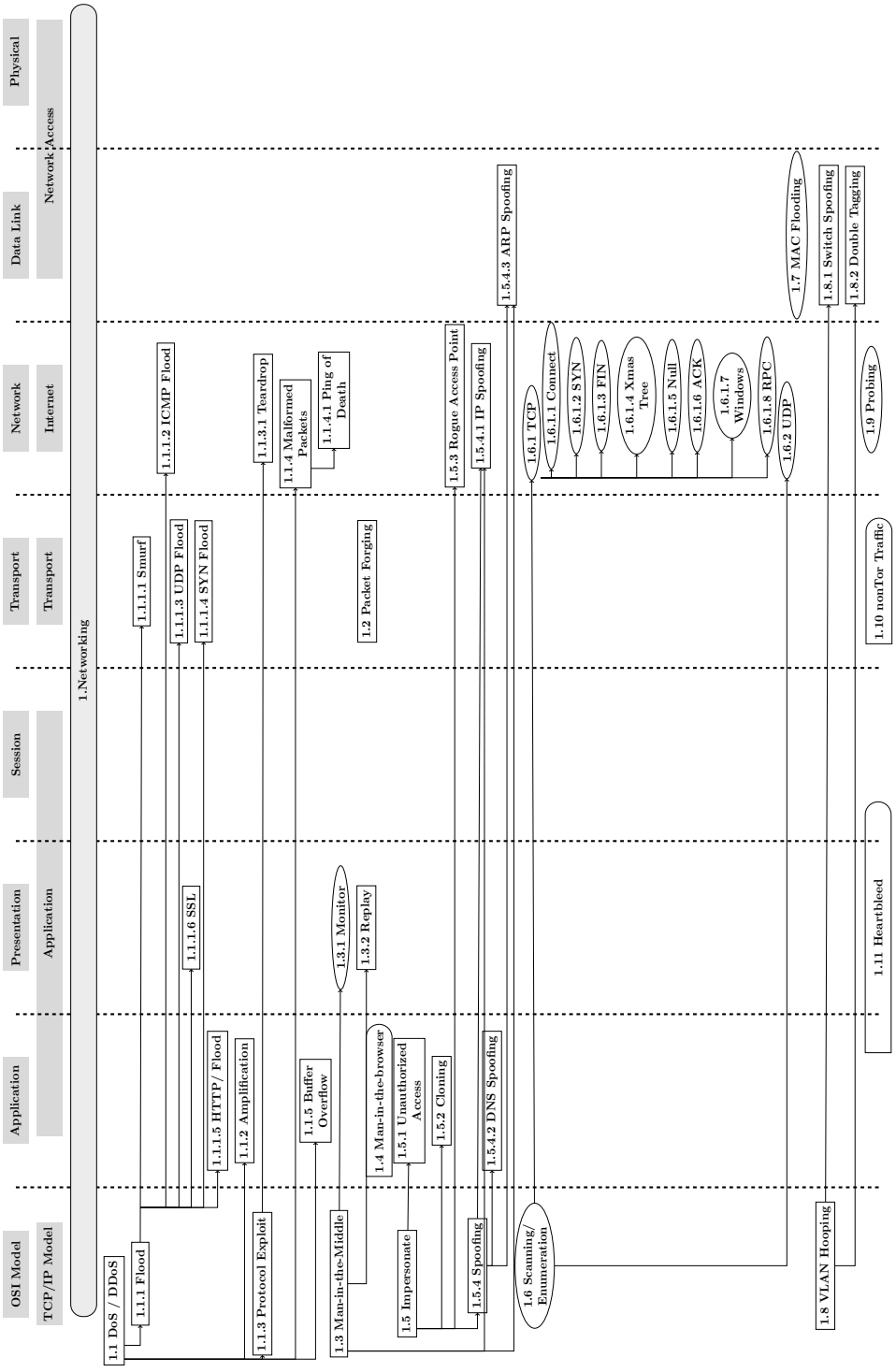


Figure 3.3
Threats Taxonomy (1 of 3)

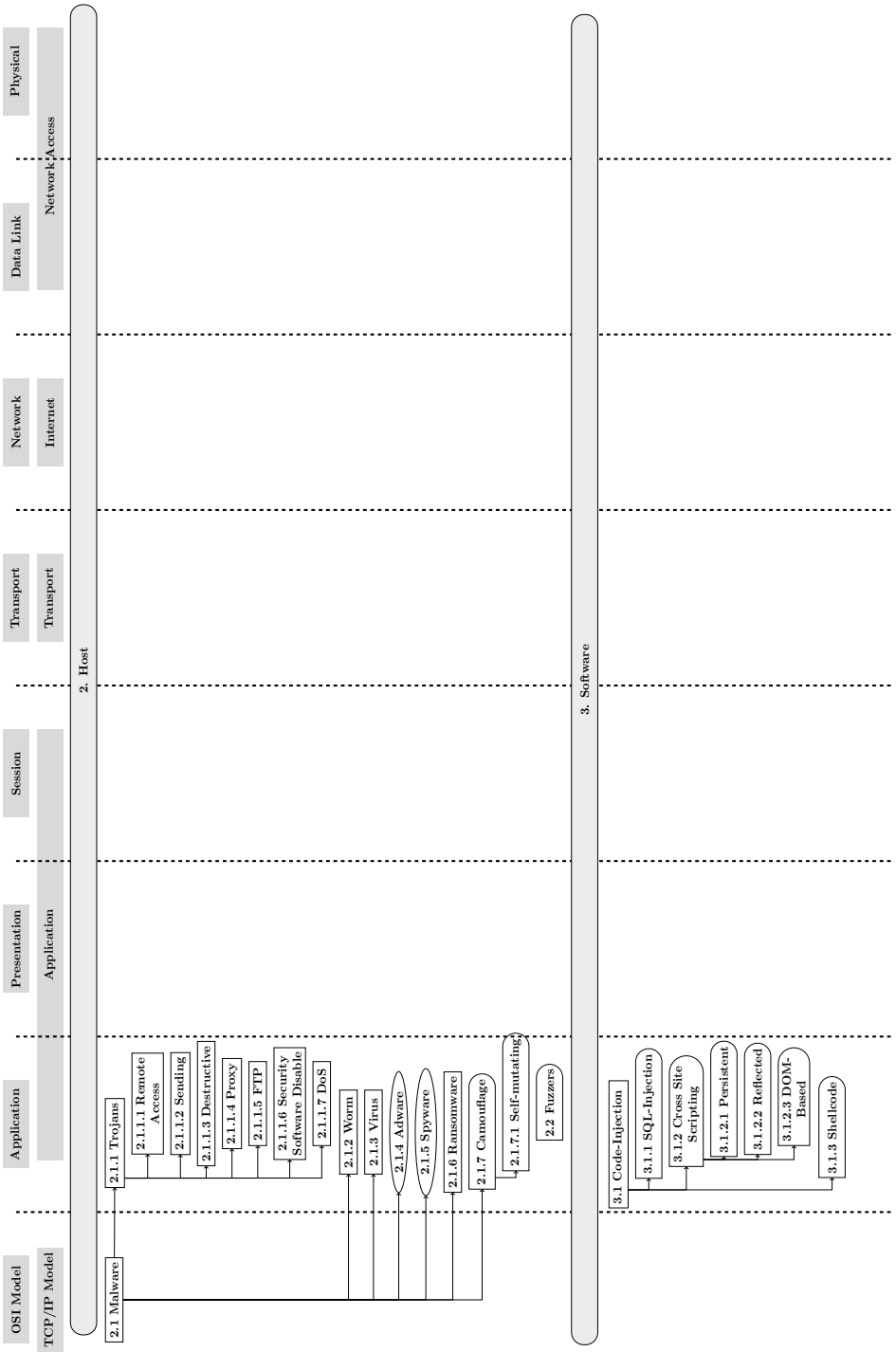


Figure 3.3
Threats Taxonomy (2 of 3)

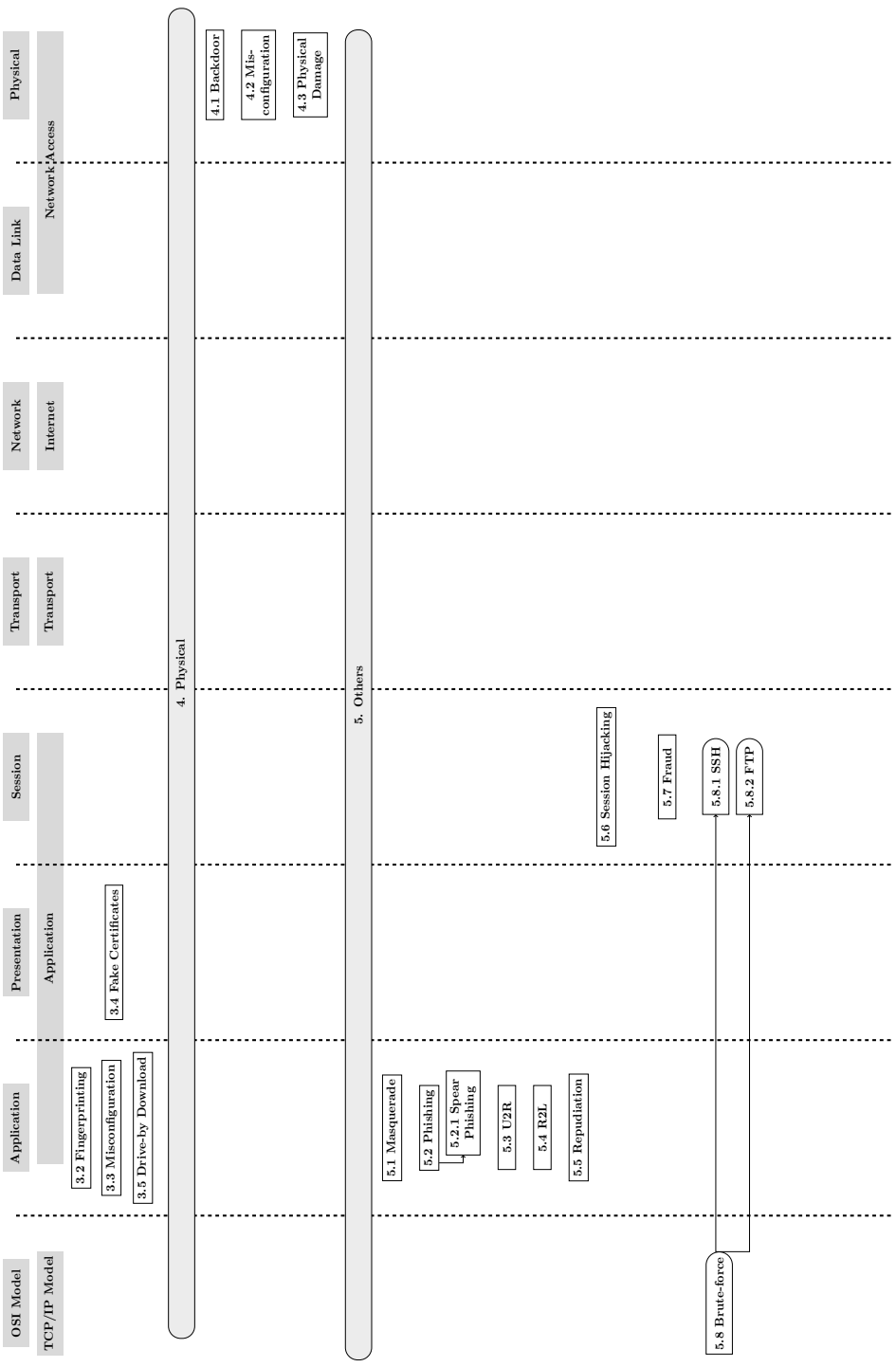


Figure 3.3
Threats Taxonomy (3 of 3)

for a certain service or website occurs. This arises immediately upon the occurrence of a significant event, such as breaking news, sales events, etc.

DoS and DDoS are divided into four categories; flood attacks (Figure 3.3 - 1.1.1), amplification attacks (Figure 3.3 - 1.1.2), protocol exploit (Figure 3.3 - 1.1.3), and malformed packets (Figure 3.3 - 1.1.4). These are defined respectively through attack examples. Smurf attack (Figure 3.3 - 1.1.1.1) generates a large number of ping requests and aims to exploit network characteristics. Internet Control Message Protocol (ICMP) Flood (Figure 3.3 - 1.1.1.2) is similar to Smurf attack since it floods the network with ICMP echo requests (ping requests). In a similar manner, User Datagram Protocol (UDP) flood (Figure 3.3 - 1.1.1.3), SYN flood (Figure 3.3 - 1.1.1.4) and HTTP flood (Figure 3.3 - 1.1.1.5) initiate a DoS attack by overwhelming the network with UDP packets targeting random ports, a huge number of TCP SYN requests and HTTP GET and POST requests, respectively. Finally, SSL attack exhausts the network by sending useless SSL data or abusing SSL handshake. Based on Neustar's Security Operations Centre report, DDoS attacks increased by 151% in the first quarter of 2020 compared with 2019 [193].

The Teardrop (Figure 3.3 - 1.1.3.1) attack takes place when an incorrect offset is set by the attacker. The ping of Death (Figure 3.3 - 1.1.4.1) attack occurs when packets are too large for routers and splitting is required. Buffer Overflow (Figure 3.3 - 1.1.5) occurs when a program writes more bytes than allowed. This occurs when an attacker sends packets larger than 65536 bytes (allowed in the IP protocol) and the stack does not have an appropriate input sanitation in place.

Packet forging (Figure 3.3 - 1.2) is the second networking attack in the presented taxonomy. Packet forging or injection occurs when an attacker generates packets that mimic normal network traffic. These generated packets can be used to perform unauthorised actions and steal sensitive data like login credentials, personal data, credit card details, Social Security Number (SSN), etc.

During a Man in the Middle attack (Figure 3.3 - 1.3), an attacker monitors or intercepts the communication between two or more nodes on the network. The attack can be passive or active when controlling the communication. On the other hand, a Man in The Browser attack (Figure 3.3 - 1.4) intercepts the browser to alter or add fields to a web page. The added fields intend to plunder confidential data, for example, by asking the user to enter sensitive information.

Impersonation (Figure 3.3 - 1.5), or pretending to be another user, takes different forms. An attacker can impersonate a user to gain higher security level and acquire access to unauthorized data (Figure 3.3 - 1.5.1) or perform cloning (Figure 3.3 - 1.5.2). Cloning is a common attack in social networks to impersonate an individual to leverage information. One type of impersonation in wireless networks is Rogue access points (Figure 3.3 - 1.5.3). During an IP spoofing attack (Figure 3.3 - 1.5.4.1), an attacker spoofs an IP address and sends packets impersonating a legitimate host. Domain Name System (DNS) spoofing, also known as DNS cache poisoning, (Figure 3.3 - 1.5.4.2) is another type of spoofing attack. An attacker, in this case, attempts to redirect packets by poisoning the DNS. Finally, Address Resolution Protocol (ARP) spoofing (Figure 3.3 - 1.5.4.3) is used to perform attacks like Man in the Middle, in order to dissociate legitimate IP and Media Access Control (MAC) addresses in the victims' ARP tables.

Scanning/enumeration is an essential step for initiating different attacks. To perform a scanning attack (Figure 3.3 - 1.6), an attacker starts to search the network for useful information such as active nodes, running OS, software versions, etc. As defined in [194], scanning has many forms, using different protocols such as Transmission Control Protocol (TCP) (Figure 3.3 - 1.6.1) or UDP (Figure 3.3 - 1.6.2). Scanning and enumeration fall under "Information Gathering".

MAC address flooding (Figure 3.3 - 1.7), and Virtual Local Area Network (VLAN) hopping (Figure 3.3 - 1.8) are also networking attacks. In MAC flooding (Figure 3.3

- 1.7), the attacker targets the network switches and as a result, packets are redirected to the wrong physical ports, while the VLAN hopping attack has two forms of either switch spoofing (Figure 3.3 - 1.8.1) or double tagging (Figure 3.3 - 1.8.2).

The last three networking attacks, in the presented taxonomy, are Probing (Figure 3.3 - 1.9), nonTor Traffic (Figure 3.3 - 1.10), and Heartbleed (Figure 3.3 - 1.11). During a probing attack, an attacker is actively footprinting a system for vulnerabilities. In Tor networks, nonTor traffic is considered anomaly. Finally, Heartbleed is an attack based on a bug in the OpenSSL library.

3.2.2 Host Threats

Host attacks, unlike networking attacks, target a specific host or system. The attack is conducted by running malicious software which aims to compromise or corrupt system functionalities. Host attacks are categorised under the malware (Figure 3.3 - 2.1) category which includes Trojans (Figure 3.3 - 2.1.1), worms (Figure 3.3 - 2.1.2), virus (Figure 3.3 - 2.1.3), adware (Figure 3.3 - 2.1.4), spyware (Figure 3.3 - 2.1.5), ransomware (Figure 3.3 - 2.1.6) and camouflage attacks (Figure 3.3 - 2.1.7).

Trojans contribute to 51.45% of all malware [195] and they often look like trusted applications but allow an attacker to control a device. Viruses affect programs and files when shared with other users over the network, whilst worms are known to self-replicate and affect multiple systems. Adware is known for showing advertisements to users when surfing the Internet or installing software. Although adware is less likely to run malicious code, it can compromise the performance of a system. Spyware monitors and tracks user actions or gathers information such as documents, user cookies, browsing history, emails, etc.

Ransomware is a relatively new type of malware, which rose 350% in 2018 [195], where the system is kept under the control of an attacker - or a third entity. This is done through encrypting files until the user or the organisation pays a ransom. The encryption key is then released and the files are recovered [196]. Finally, camouflage malware evolved over time reaching polymorphic and metamorphic techniques in 1990 and 1998, respectively [197, 198]. For example, self-mutating malware could use numerous techniques, such as instruction substitution or permutation, garbage insertion, variable substitutions, and control-flow alteration [199].

3.2.3 Software Threats

Software threats are grouped in the Code injection (Figure 3.3 - 3.1) category in which an attacker "injects" malicious code that affects the execution path of a certain program or system. This category includes Structured Query Language (SQL) Injection, during which an attacker attempts to inject a query to a target database. This query could result in obtaining confidential data or deleting data by dropping columns, rows, or tables. Cross Site Scripting (XSS), as another type of code injection attacks is used to run malicious code in a web application to steal cookies or credentials. XSS has three main categories. In persistent/stored XSS (Figure 3.3 - 3.1.2.1), a script is saved to a database and is executed every time a page is loaded. In Reflected XSS (Figure 3.3 - 3.1.2.2), the script is part of an HTTP request sent to the server. Document Object Model (DOM)-based XSS (Figure 3.3 - 3.1.2.3) is considered an advanced type of XSS where the attacker changes values in the DOM e.g., document location, document Uniform Resource Locator (URL), etc. DOM-based XSS is difficult to detect as the script is never transferred to the server.

Fingerprinting (Figure 3.3 - 3.2) and misconfiguration are also forms of software threats. Fake server certificates (Figure 3.3 - 3.4) are considered alarming and should be considered while analysing communications as they could deceive the browser/user

thinking that the connection is secure. This could result in phishing websites looking legitimate. Moreover, they could be used as a seed to perform other attacks like Man-in-the-Middle.

Finally, Drive-by or download (Figure 3.3-3.5) is another software threat that requires no action from the user, however, the malicious code is automatically downloaded. In 2017, it contributed to 48% of all web-based attacks [200, 201] and is considered one of the main threats in 2019 [202].

3.2.4 Physical Threats

Physical attacks are a result of a tampering attempt on the network hardware (edge, or other devices) or its configuration. This can include changing configurations (Figure 3.3 - 4.2) and introducing backdoors (i.e., The Evil Maid). CI and IoT networks are usually exposed to physical threats. It is important to note that physical threats can also include physical damage (Figure 3.3 - 4.3).

3.2.5 Other Threats

The last category of threats contains miscellaneous threats. Most of these threats are influenced by the attacker actions. This category includes user masquerade (Figure 3.3 - 5.1) in which the attacker uses a fake identity. Phishing is another form of attacks that relies on social engineering. In a phishing attack, an attacker uses emails or other electronic messaging services to obtain credentials or confidential data. Spear phishing, unlike phishing, targets a specific user or organisation. The attacker, in this case, pretends to own and/or know specific details and personal data.

When the attacker attempts to gain authorised access or higher privileges to the target system, either by promoting to a root user or gaining local access, these attacks are called U2R (Figure 3.3 - 5.3) and R2L (Figure 3.3 - 5.4), respectively.

Additionally, a user can be denied an action such as repudiation attack (Figure 3.3 - 5.5). Human attacks can also include session hijacking (Figure 3.3 - 5.6) or sniffing. These attacks are based on the attacker gaining access over an active session to access cookies and tokens. Finally, brute-force attacks (Figure 3.3 - 5.8), either Secure Shell (SSH) (Figure 3.3 - 5.8.1) or File Transfer Protocol (FTP) (Figure 3.3 - 5.8.2), are another form of human threats. Attackers in this case attempt to authenticate by trying various passwords or passphrases.

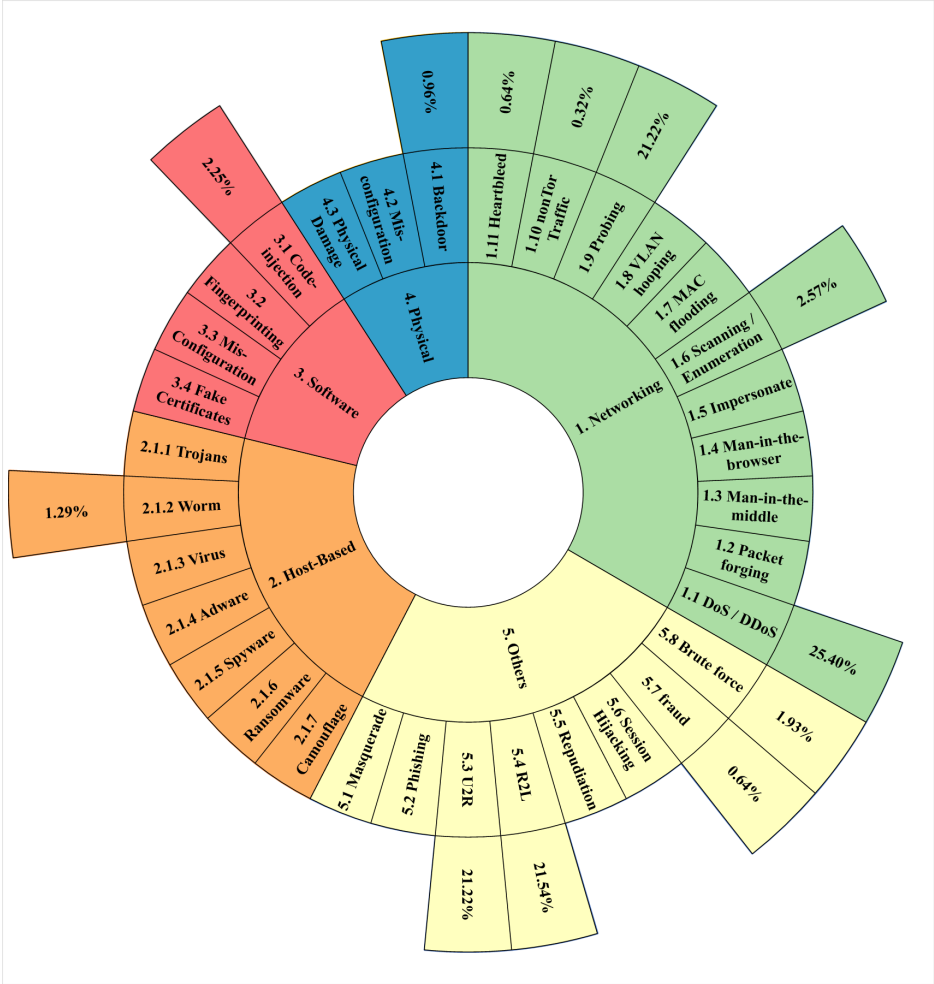
3.3 Attacks Coverage

Based on the taxonomy discussed in Section 3.2 and the recent IDS articles outlined in Table 3.1, it can be observed that some attacks are in the focal point of research while a lot are not considered by recent IDS. This is due to the underrepresentation of these attacks in recent datasets and the difficulties associated with generating datasets.

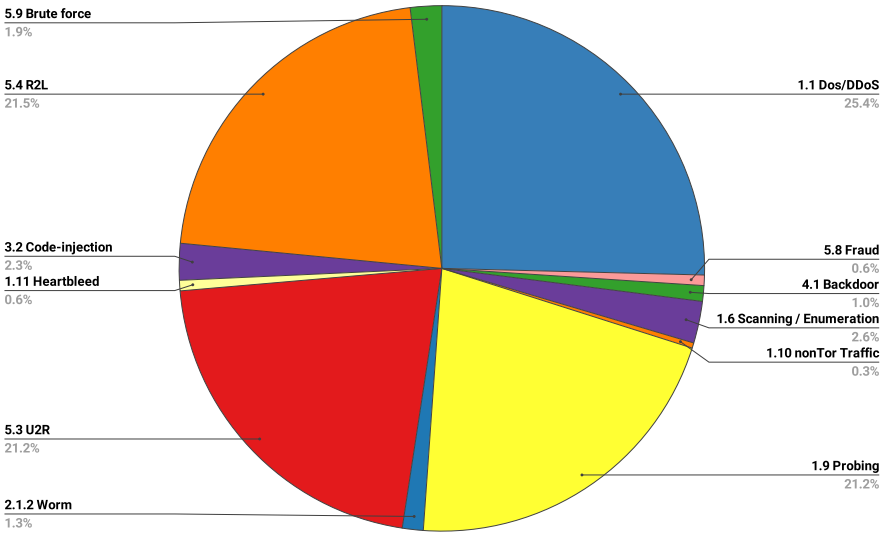
Figure 3.4a visualises all the threats presented in the taxonomy. The percentages in the outer circle represent attacks covered by the IDS discussed in Table 3.1. As shown, a large number of attacks (63.6%) are uncovered (i.e., not represented in the recent datasets, thus not detected by IDS).

Figure 3.4b visualises the attacks detected by the different IDS presented in Table 3.1. It is shown that the four attacks available in the KDD Cup'99 dataset are the most used by IDS research, namely; DoS/DDoS, Probing, R2L and U2R. It is also noted that these same four attacks are the ones available in the NSL-KDD dataset and the DARPA datasets. The popularity of the KDD dataset family and their readiness for ML development contributes to this skewness in the attacks detected by recent IDS.

Only 12 attack categories from the presented taxonomy are listed in Figure 3.4b which highlights potential limitations of these IDS to cope with the broad range of attacks and zero-day attacks. To tackle the detection of zero-day attacks, there is a



(a) Occurrence of all attacks categories based on the presented taxonomy



(b) Distribution of Attacks discussed in Table 3.1

Figure 3.4
Distribution of Covered Attacks in IDS from Articles Listed in Table 3.1

need to build extendable datasets that could be used to train different ML models used for detection. By employing extendable datasets and a standardised method for dataset generation, alongside advancements in ML [203, 204], new attacks can be integrated into anomaly-based datasets and consequently, utilised by IDS.

3.4 Summary

In this chapter, recent IDS articles are analysed and discussed. The analysis shows the wide range of algorithms used to build IDS. Table 3.1 lists the datasets and algorithms used by each of the articles, while Figure 3.2 plots the commonly used algorithms and their frequency of use. The discussion highlights (i) the absence of the representation of new attacks in IDS datasets. (ii) The lack of datasets for intrusion detection on special-purpose networks, like IoT, limiting the availability of suitable deployable IDS, and (iii) the dominance of ML usage to build anomaly-based IDS. Due to the pace at which new cyber attacks are rising, new non-traditional ML techniques are needed to build appropriate IDS that can learn from limited data and are capable of detecting zero-day attacks. To further demonstrate the gap, a cyber threats taxonomy is presented. The taxonomy classifies cyber threats based on the OSI layer, active or passive behaviour and threat source. Although comprehensive, the presented taxonomy is built in an extendable fashion and is publicly available for future amendments. The presented taxonomy confirms the cyber attack representation gap.

Based on the analysis covered in this chapter, the next chapter addresses the highlighted gap in special-purpose network datasets and IDS. This is done by investigating the classification of anomalies and cyber attacks in two case studies of special-purpose networks; SCADA and IoT using six ML techniques.

Chapter 4

Utilising Machine Learning for Special-Purpose IDS

4.1 Problem Statement

The lack of datasets for special-purpose networks (i.e., SCADA, Industrial Automation and Control Systems (IACS), Distributed Control Systems (DCS), and IoT networks was highlighted in Chapter 3. This dataset shortage directly affects the advancement of IDS in this regard. This is due to the reliance of research on dataset availability for analysis and training up-to-date IDS. Furthermore, with the increased dependence on automation and advancement in deployed solutions, current CI and IoT systems are vulnerable to faulty operations and cyber attacks [205].

Robert Mitchell and Ing-Ray Chen [206] survey recent IDS for Cyber-Physical Systems (CPS) and CI usage. The authors classify IDS based on detection techniques into knowledge-based and behaviour-based, and based on audit into host-based and network-based, which aligns with the IDS classification previously discussed. The authors analyse aerospace, automotive, medical, and SCADA IDS. Their analysis shows that, out of the 32 IDS papers they considered, 22 (68.75%) do not release their

dataset, 4 (12.5%) do not report the dataset used and 4 (12.5%) use public datasets. Two of the four public datasets are KDD Cup'99 dataset, NSL-KDD dataset, which are general-purpose IDS datasets. The other two datasets cover replay and unauthorisation attacks.

Amin *et al.* [207, 208] overview the various security threats of CI networks which include threats targeting different layers (physical, regulatory control and supervisory). This emphasises the different elements and requirements of CI networks. Furthermore, Cheng *et al.* [209] highlight the lack of available mechanisms for CI IDS and Mathur [210] discusses the challenges facing the detection of anomalies and incidents in CI. These challenges include failing to detect coordinated cyber attacks and high FPR that, based on the author's investigation, are beyond the acceptable range.

Current general-purpose IDS fall short in delivering the security needs for special-purpose networks. This is reasoned by multiple factors that include, but not limited to, the specific requirements and architectures of these systems, the heterogeneity of legacy protocols, their scale, computational power, and uniqueness of the usage scenarios [10]. Therefore, building IDS that can cope with these requirements is a pressing need. However, the limited availability of public IoT and CI datasets often form a barrier against this advancement.

To tackle this problem, this chapter focuses on building special-purpose IDS while exploring the different challenges that accompany this process. Six ML algorithms and two real-world datasets are used for evaluation. These algorithms are the most commonly used in the literature [211]. The first dataset is generated by the French Naval Academy that simulates a CI that controls a water SCADA system [81]. The dataset comprises real-world scenarios that cover hardware failures, sabotage, and cyber attacks. The second dataset is generated and collected using a simulated IoT network that is based on MQTT protocol [212]. The dataset comprises normal operations and four cyber attack scenarios.

Firstly, this chapter provides an overview of the six different ML techniques that are used to build the IDS models. Secondly, the first dataset is outlined (SCADA dataset) where the dataset properties, scenarios description, and dataset preprocessing are presented. Later, the SCADA IDS models are explained and the experimental results are discussed. Thirdly, an MQTT simulated network is used to generate a novel MQTT-IDS dataset (MQTT-IoT-IDS2020 dataset). The dataset covers benign traffic, general brute-force cyber attacks, and MQTT-based attacks. The dataset collection and processing are described. Three levels of feature abstraction are established; namely, packet-based, unidirectional flow, and bidirectional flow. The generated dataset is then used to train and evaluate ML-based IDS. Finally, the chapter is summarised to show the IDS performance and limitations.

4.2 Background

The following subsections introduce the six ML techniques that are used in this chapter to build IDS for CI and IoT networks. These techniques are Logistic Regression (LR), Naïve Bayes (NB), k-NN, SVM, DT, and Random Forest (RF) [213, 214]. These are amongst the commonly used techniques in the literature for evaluating IDS as outlined in [43, 215, 211]. Furthermore, the analysis of recent IDS articles presented in Chapter 3 demonstrates that these six are the most used succeeding ANN with 16.4%, as follows: SVM (13.23%), DT (11.11%), NB (10.05%), k-NN (5.82%), RF (4.76%), and LR (3.17%). The use of different techniques allows a comparison and analysis of their varying performance. Finally, given that special-purpose networks datasets have a small number of features and instances, the chosen techniques are suitable for the required analysis.

4.2.1 Logistic Regression

LR is a well-established statistical technique for classification [216]. The model is based on the logistic, or sigmoid function (Equation 4.1), and the training goal is to fit the function to best split the training data. In a 2D space, the resulting curve can be visualised as S-shape as shown in Figure 4.1.

$$f(x) = \frac{1}{1 + e^{-x}} \tag{4.1}$$

LR can be (i) binary, where the dependant variable (i.e. the output) is a class of two possible options (e.g., benign and anomaly), (ii) multinomial, where the dependant variable can be drawn out of many classes (e.g. benign, attack 1, and attack 2) or (iii) ordinal, which is multinomial except that the classes have an ordinal relation (e.g., attack severity) [217].

The output of LR is determined based on a decision boundary and a threshold. In the binary case, for example, if the output is ≥ 0.5, it belongs to class A, otherwise, it belongs to class B, as shown in Equation 4.2.

$$Y = \begin{cases} A, & if(f(x) \geq 0.5) \\ B, & otherwise \end{cases} \tag{4.2}$$

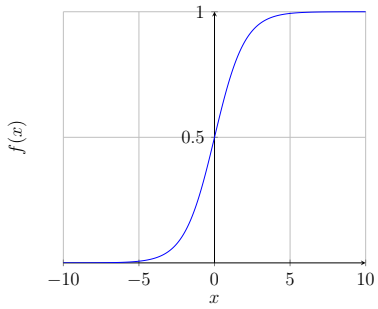


Figure 4.1
LR Sigmoid Function

4.2.2 Naïve Bayes

NB classifier relies on Bayes theory [218] which depends on the conditional probability defined as "The likelihood of an event or outcome occurring, based on the occurrence of a previous event or outcome" [219]. Conditional probability $p(C_i|x)$, where x represents the input and C_i is the i^{th} class, determines how likely an instance belongs to class i. Based on Bayes theory; $p(C_i|x)$ is expanded as shown in Equation 4.3, which resembles the prior probability (class) multiplied by the likelihood divided by the evidence. This technique is "naïve" as it assumes that all features (x_1, x_2...x_n) are mutually independent of each other.

$$p(C_i|x) = \frac{p(C_i) \times p(x|C_i)}{p(x)} \tag{4.3}$$

Given n features that represent x, the likelihood ($p(x|C_i)$ of each feature is calculated based on its occurrence in the training data, and is determined by Equation 4.4.

$$p(x|C_i) = p(x_1|C_i) \times p(x_2|C_i) \times p(x_3|C_i) \times .... \times p(x_n|C_i) \tag{4.4}$$

After training, the class label of x is determined based on the maximum probability $max(p(C_i|x))$. It is important to note that $p(x)$ is used in Bayes theory as a normalisation term in order to calculate the probability. Without $p(x)$, the output of $p(C_i|x)$ does not represent a probability, however, since the NB technique aims to decide which class label to assign to an unknown instance, the probabilistic numeric value is insignificant, therefore $p(x)$ can be dropped to reduce computations and only $p(C_i) \times p(x|C_i)$ is used.

If the features of x are continuous, it is assumed that they follow a normal distribution (Gaussian distribution). Thus, the probabilities are calculated based on the mean (μ) and the standard deviation (σ) of the training data occurrences as shown in Equation 4.5.

$$p(x_j|C_i) = \frac{1}{\sqrt{2 \times \pi \times (\sigma_{x_j,C_i})^2}} \times exp(-\frac{(x_j - \mu_{x_j,C_i})^2}{2 \times (\sigma_{x_j,C_i})^2}) \tag{4.5}$$

4.2.3 k-Nearest Neighbour

k-NN is known as one of the most popular classifiers due to its effectiveness and simplicity [220]. k-NN is based on distance measurement. Using the training instances, a new instance is classified based on its similarity to the training instances. Specifically, the closest k instances to the new one are the ones that determine the classifier's decision [214].

The distance between two instances x and y can be calculated using various formulas. For example, Equation 4.6 is used to calculate the L2 norm (Euclidean distance), while Equation 4.7 for the L1 norm (Manhattan distance).

$$||x||_2 = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{4.6}$$

$$||x||_1 = \sum_{i=1}^{n}(|x_i - y_i|) \tag{4.7}$$

The distances between a new instance and all instances in the training data are calculated and sorted. The least k distances are used to decide to which label the new instance belongs, using majority voting. It is important to mention that if k is too small (1 for example), the model will be sensitive to noisy inputs and will not be able to generalise [221]. Figure 4.2 show sample k-NN boundaries, where two classes are plotted with red and blue circles, and an unknown instance is plotted in yellow. Based on the instances in Figure 4.2, the label of the unknown instance is decided as follows; with $k = 1$ and $k = 2$, the instance is labelled as Class B (Blue), while it is labelled as Class A (Red), with $k = 5$.

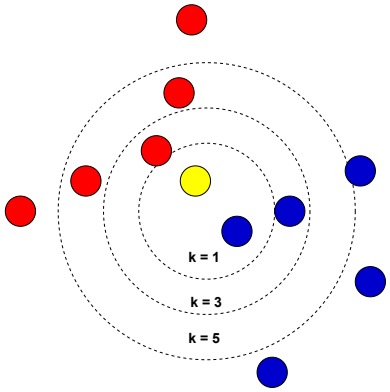


Figure 4.2
k-NN Sample [222]
Class A (Red), Class B (Blue), and Unknown Instance (Yellow)

4.2.4 Support Vector Machine

SVM is one of the well-established supervised ML techniques [223]. Given the training samples, SVM training goal is to construct a hyperplane in a high-dimensional space that best separates the given classes [224]. Formally, given two classes, the minimisation problem of SVM is represented as follows [225].

$$\min_{\mathrm{w}\in\mathbb{R}^d} \|\mathrm{w}\|^2 + C\sum_{i}^{N} \underbrace{\max(0, 1 - y_i f(x_i))}_{\text{Loss}}$$

$$f(x_i) = (\mathrm{w}^T x_i + b)$$

Where C is a regularisation parameter to represent the trade-off between ensuring that x_i is on the expected side of the plane and increasing the margin. If an SVM is working in a two-dimensional space, then the hyperplane is visualised as a line. In a three-dimensional space, it is a plane, and an n-dimensional plane when working in higher dimensions.

A data point falls in one of three places in relation to the hyperplane based on $y_i f(x_i)$. If $y_i f(x_i)$ is greater than 1, then the point is outside the margin and does not contribute to the loss. If $y_i f(x_i)$ equals 1, then the point is on the margin. Finally, if $y_i f(x_i)$ is less than 1, then the point contributes to the loss as it is on the wrong side [226].

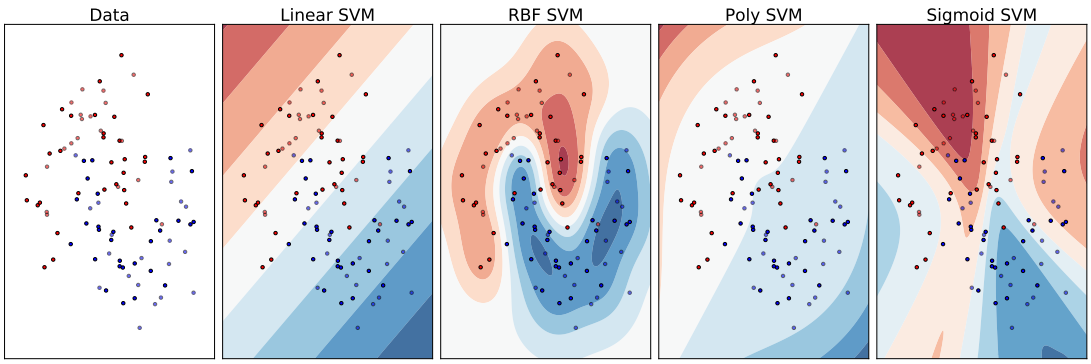


Figure 4.3
SVM Samples (Linear, RBF, Polynomial (3rd degree), and Sigmoid) [227]

When the data is not linearly separable, SVM use a kernel to map the data (input features) to a nonlinear higher dimensional space in which a hyperplane best separates the classes. SVM kernels include linear, Radial Basis Function (RBF), polynomial, and sigmoid [211]. SVM kernels and decision boundaries are visualised in Figure 4.3.

4.2.5 Decision Tree and Random Forest

DT are composed of a group of branches that represents feature-based tests [213, 228]. Given a decision tree root, the tree starts branching based on the feature values until reaching a leaf where the decision is made (i.e., the class is determined), as shown in Figure 4.4.

During the training process, the training data is recursively split until a decision tree is built. To reduce the complexity of DT and avoid overfitting, pruning process takes place. Pruning removes redundant and noncritical branches, which leaves the tree more sparse.

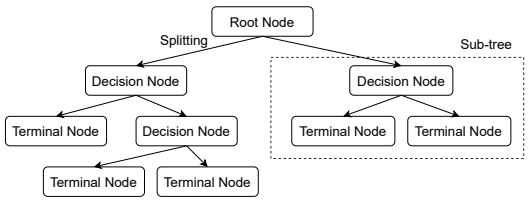


Figure 4.4
Decision Tree Sample [211]

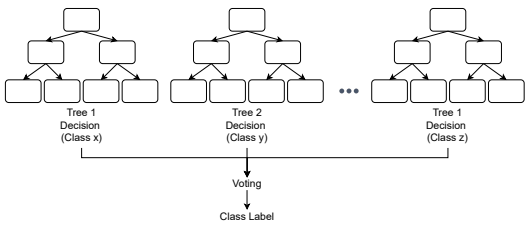


Figure 4.5
Random Forest Sample [211]

RF, on the other hand, are a group of DT, as shown in Figure 4.5. The output of the DT is combined to reach a classification decision in an ensemble fashion. Bootstrapping, also known as Bagging [228], is a statistical technique used to split the data and features among different DT.

Table 4.1 provides a short summary of the advantages and disadvantages of each of the ML algorithms discussed in this section. Liu and Lang [43], Xie *et al.* [229], and Mishra *et al.* [211] survey different ML techniques, their characteristics, pros, and cons when they are applied to IDS. The authors highlight that these techniques are the most used, which aligns with the analysis presented in Chapter 3.

Table 4.1
ML Techniques Summary

Algorithm	Advantages	Disadvantages
LR	- Simple to understand and implement - Fast training	- Low performance with large feature space - Classes separability assumption
NB	- Simple to understand - Fast to classify - Scalable and can learn from small dataset	- Poor performance when training data is not representable - Assumes feature independence - Hard to operate with continuous data
k-NN	- Simple to implement - Easy to understand - No training required - Flexible in terms of choosing the function that represents distance	- Slow (curse of dimensionality) - High time/memory complexity - Does not perform well with imbalanced datasets - Sensitive to the choice of K
SVM	- Well-suited for high-dimensional data - Can work with non-linear features - Can learn from small data	- Not suited when classes overlap - Relatively slow - Computationally intensive - Choosing the kernel can be challenging
DT	- Easy to explain predictions - Features interactions are taken into account - Can train using continuous and discrete features	- Sensitive to data - Prone to overfitting
RF	- Less sensitive to data compared to DT - Performs well on large datasets - Mitigates the DT overfitting problem	- Predictions are not easy to explain - Slow training

Python v3.6.4 [230] and scikit-learn v0.21.3 [227] are used to implement the six ML techniques with the parameters as follows:

- random_state: 0
- Algorithm-specific parameters:
  - LR
    - penalty: l2
    - fit_intercept (bias): True
    - solver: lbfgs
    - max_iter: 100
  - NB
    - var_smoothing: 1e-9
  - k-NN
    - n_neighbors: 5
    - weights: uniform
    - algorithm: auto
    - leaf_size: 30
  - SVM
    - kernel: linear and rbf
    - degree: 3
    - gamma: scale
    - tol (tolerance): 1e-3
    - shrinking: True
  - DT and RF
    - criterion: entropy
    - splitter: best
    - max_depth: none
    - min_samples_split: 2
    - min_samples_leaf: 1
    - max_features: Square root of number of features
  - RF
    - n_estimators: 10

4.3 SCADA Dataset

Having discussed the problem this chapter is addressing and the ML techniques used, this section aims to overview the first dataset, which is generated from a CI water system controlled by SCADA.

4.3.1 SCADA Dataset Architecture

The CI is composed of a 9 litre main tank and a 7 litre secondary one. Each tank can either store or distribute liquid (water in this case). The main tank is equipped with four sensors connected to the Programmable Logic Controllers (PLC). Figure 4.6 shows the physical architecture of the control system, while a high-level diagrammatic representation is depicted in Figure 4.7.

The four sensors are used to measure the liquid level in the main tank. “Discrete sensor 0” indicates a low level in the tank (1.25L). “Discrete sensor 1” indicates a measure of less than 3.35L. “Discrete sensor 2”, indicates a level of 8L while “Discrete

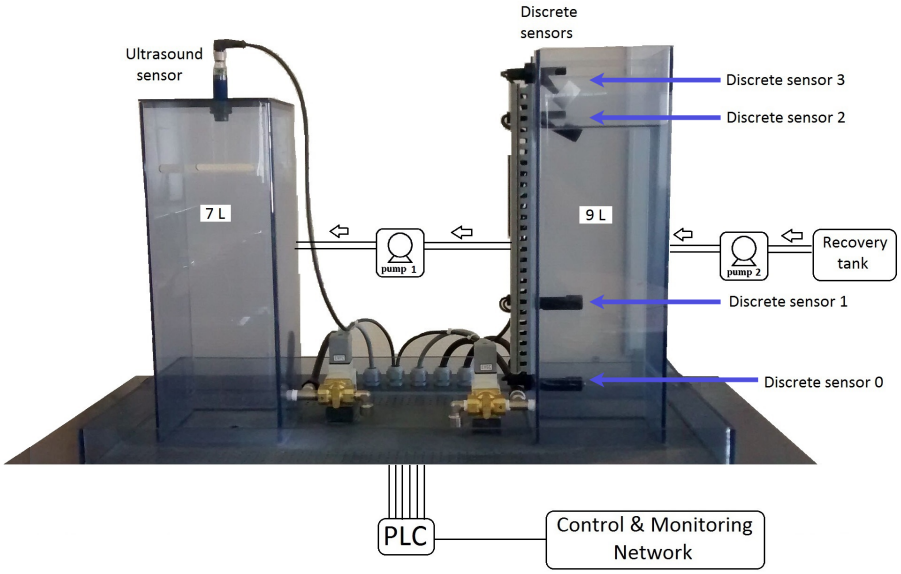


Figure 4.6
SCADA System Architecture [81]

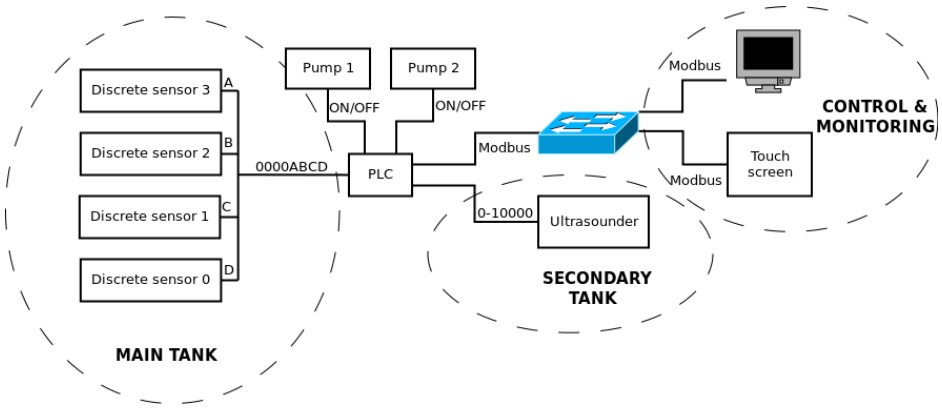


Figure 4.7
SCADA Network High-Level Architecture [81]

sensor 3" indicates a full tank measure (9L) [81]. The sensors inputs (A, B, C, and D) are concatenated as one register (0000ABCD) as shown in Figure 4.7.

There are two pumps, Pump1 and Pump2, that control the flow between the main and secondary tanks. The liquid volume in the secondary tank is monitored by an ultrasound sensor installed at the top of the tank. It measures the volume using the distance from the liquid surface to the top of the tank. It is also used to detect the existence of liquid in the tank. All sensors use the Modbus protocol to transfer the collated data to the control and monitor network.

4.3.2 SCADA Operation and Dataset Scenarios

During the dataset generation, the primary tank is filled from a recovery tank simulating a liquid source, which, in real life, can be a fuel line or a river. When the primary tank is filled (using Pump2), the PLC activates Pump1 to transfer the liquid from the primary tank to the secondary tank to avoid spillage. Constant liquid consumption is simulated using the valves at the bottom of the tanks.

The full operation is monitored by the PLC using sensory data readings recorded on interval of 0.1 seconds [81]. If the liquid volume in the primary tank goes below 1.25L, the PLC turns on Pump2 automatically to allow refilling from the recovery tank.

Table 4.2
SCADA: PLC Registers Extracted Bits Representation

Reg. No.	Bit No.	Value
2	4	Discrete Sensor 3
	5	Discrete Sensor 2
	6	Discrete Sensor 1
	7	Discrete Sensor 0
3	0	Pump2
	1	Pump1
	5	Pump2 Valve
	4	Pump1 Valve
4	16-bits	Depth Sensor

In a similar fashion, Pump2 will be turned off when a total of 9L is reached. Pump1 is automatically activated when the ultrasound sensor detects a liquid level below 2.1L and automatically deactivates when the liquid reaches 6.3L.

The dataset consists of Comma-Separated Values (CSV) format files that contain the sensor readings, captured from the PLC registers (2, 3, and 4) to describe the state of the system. Table 4.2 provides an overview of the different registers and their corresponding usage. Register2 bits provide the binary state of each discrete sensor. To retrieve each separate sensor value, bitwise masks and operations are used. Register3 indicates the state of the two pumps and valves showing whether each is activated or deactivated. Register4 contains the depth sensor reading represented as a step value from 0 to 10,000 for the ultrasound sensors (e.g., step 3,000 represents 2.1L of liquid in the tank).

The dataset comprises 14 different scenarios besides normal behaviour. These are listed in Table 4.3, with each scenario covering one of five operational scenarios representing potential threats (i.e., sabotage, accident, breakdown, or cyber attack) as well as six affected components. The affected components are system components that are instantly affected by the anomaly. The dataset is provided in 15 CSV files; one for each scenario.

Table 4.3
SCADA: Dataset Scenarios, Operational Scenarios, and Affected Components

	Scenario	Affected Component	Operational Scenario	No. of instances
1	Normal	None	Normal	5519
2	Plastic bag	Ultrasound Sensor	Accident/ Sabotage	10549
3	Blocked measure 1		Breakdown/ Sabotage	226
4	Blocked measure 2			144
5	Floating objects in main tank (2 objects)		Accident/ Sabotage	854
6	Floating objects in main tank (7 objects)			733
7	Humidity		Breakdown	157
8	Discrete sensor failure	Discrete sensor 1		1920
9	Discrete sensor failure	Discrete sensor 2		5701
10	DoS	Network	Cyber attack	307
11	Spoofing			10130
12	Wrong connection		Breakdown/ Sabotage	6228
13	Person hitting the tanks (low intensity)	Whole subsystem	Sabotage	347
14	Person hitting the tanks (medium intensity)			281
15	Person hitting the tanks (high intensity)			292

4.3.3 SCADA Dataset Preprocessing

Figure 4.8 summarises the preprocessing steps applied to the SCADA dataset. As demonstrated, the preprocessing is composed of six stages.

1. **Extracting Instances**

   The dataset is provided in raw CSV log files, where the raw readings are recorded line by line. Each instance is represented in 10 rows, each row containing date, time, register number, and register reading/value of the PLC. At this stage, each scenario instances are extracted from the corresponding log file. An instance is represented by the PLC recording of the register values at a specific time.

2. **Calculating the rate of change of Register4**

   Register4 monitors the liquid level in the secondary tank and its value is crucial for each instance. Register4 value is demonstrated as the most significant, however, its significance does not lie in the reading value itself, but in the drift of values over time (i.e. the change trend/rate). Figure 4.9 visualises Register4 rate of change for each of the scenarios.

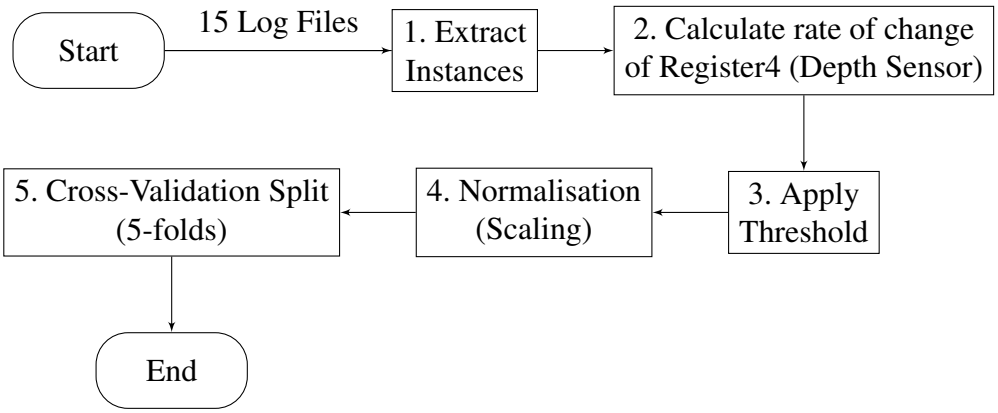


Figure 4.8
SCADA: Preprocessing Stages

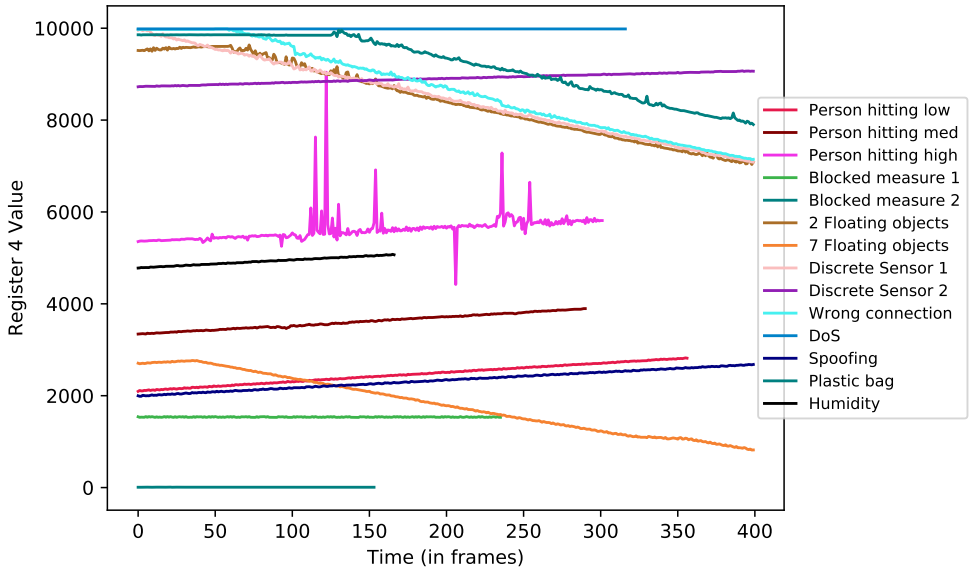


Figure 4.9
SCADA: The trends of the Rate of Change of Register4 readings for different scenarios

For each instance, the rate of change is calculated over 10 time frames as expressed in Equation 4.8.

$$Rate\ of\ change_i = \frac{reg4_i - reg4_{i-10}}{time_i - time_{i-10}} \tag{4.8}$$

3. **Applying Threshold**

   Table 4.3 demonstrates the variance in the number of instances over the different scenarios. The instances are not evenly distributed over the scenarios. Therefore, the scenario(s) with the most instances will bias the model training, thus affecting the classification output. A threshold is applied to take only the first N instances of each file. N should satisfy two conditions: (i) reduce the gap between instances count across scenarios and (ii) maintain the variation of instances per scenario.

4. **Normalisation**

   Normalisation is an essential step to ensure the features are in the same scale and ready for ML usage. Min-Max normalisation is used [39].

5. **Cross-Validation**

   Finally, the data is split into training and testing sets. A split of 80% for training and 20% for testing [231] is used over 5-fold cross-validation.

4.4 SCADA Experiments and Results

In this section, three different experiments are outlined and evaluated showing how accurately anomalies are detected. The aim of the different experiments is to provide different levels of information regarding the occurrence of an anomaly. This varies from merely reporting the occurrence of an anomaly, to identifying the affected component and the anomalous scenario.

4.4.1 Experiment 1: Anomaly Detection

In real situations, IDS should fire an alert when an anomaly occurs. The first experiment evaluates the ability of the six ML algorithms discussed earlier to flag anomalies in the SCADA dataset. The models are utilised as binary classifiers to distinguish anomalies from benign behaviour, hence this experiment does not specify the anomaly type or associated affected component as the case in the following ones.

Figure 4.10 shows the classification accuracies of the different ML algorithms used. As demonstrated, the highest accuracies reached are 94.12%, 93.67%, 93.30%, and 91.99% when using k-NN, RF, DT, and SVM with RBF kernel respectively.

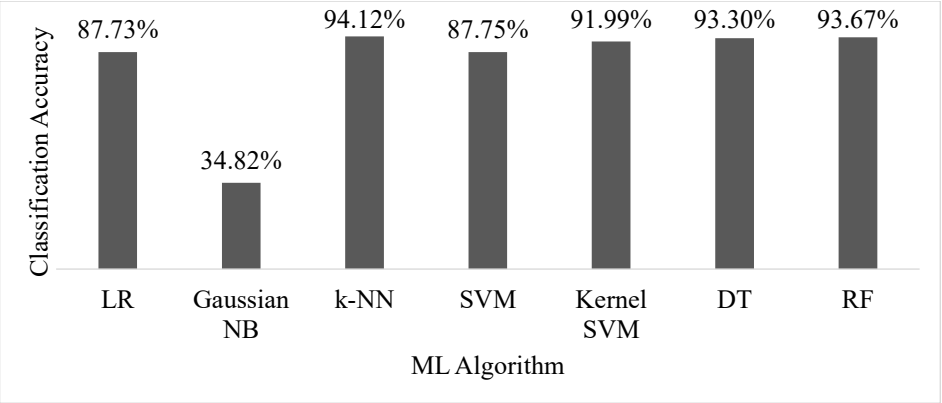


Figure 4.10
SCADA: Anomaly Detection Overall Accuracy (5-fold cross-validation)

The recall, precision, and F1-Score of the first experiment are detailed in Table 4.4. The three techniques with top overall accuracies (k-NN, DT, and RF) are able to classify both benign behaviour and anomalies efficiently. The recall of the benign class is 74.01%, 74.01%, and 75.66% and the anomaly class is 97.15%, 96.22%, and 96.38% when using k-NN, DT, and RF, respectively. The reason these three algorithms outperform the rest is due to the fact that they can map complex relations and non-linearity compared to the others that aim to fit a hyperplane or a probabilistic relation. It is important to note that in a CI setup, normal operations and anomalies can overlap [232], thus complicating the classes separability task.

Table 4.4
SCADA Results: Experiment 1 - Anomaly Detection (5-fold cross-validation)

Classification (Is Anomaly)	**Recall**	**Precision**	**F1-Score**
LR			
Benign	7.15%	90.34%	13.22%
Anomaly	99.89%	87.7%	93.4%
Weighted Average	87.73%	88.05%	88.05%
NB			
Benign	99.95%	16.74%	28.67%
Anomaly	24.99%	99.97%	39.98%
Weighted Average	34.82%	89.06%	89.06%
k-NN			
Benign	**74.01%**	79.7%	76.74%

Anomaly	**97.15%**	96.12%	96.63%
Weighted Average	**94.12%**	93.97%	93.97%
SVM			
Benign	7.15%	92.24%	13.23%
Anomaly	99.91%	87.7%	93.41%
Weighted Average	87.75%	88.3%	88.3%
Kernel SVM			
Benign	39.53%	98.52%	56.4%
Anomaly	99.91%	91.63%	95.59%
Weighted Average	91.99%	92.54%	92.54%
DT			
Benign	**74.01%**	74.72%	74.35%
Anomaly	**96.22%**	96.09%	96.15%
Weighted Average	**93.3%**	93.28%	93.28%
RF			
Benign	**75.66%**	75.99%	75.78%
Anomaly	**96.38%**	96.33%	96.36%
Weighted Average	**93.67%**	93.67%	93.67%

Firing an alert when an anomaly occurs is important, however, since the alert here is provided in a binary fashion, it is not straightforward – in this case – to identify the problem at first sight. Therefore, taking a corrective action is delayed. To this end, a second experiment is established with the aim of providing more information.

4.4.2 Experiment 2: Affected Component Classification

Instead of just firing an alert, the second experiment aims to report the affected component when an anomaly occurs using multi-class classifiers. Compared to the first experiment, this additional piece of information provides more details which assists in taking accelerated corrective actions.

Figure 4.11 shows the classification results of the different ML algorithms. The highest accuracies are 82.69%, 82.71%, and 81.79 using k-NN, RF and DT, respectively. The result shows a trade-off between the overall accuracy and the details

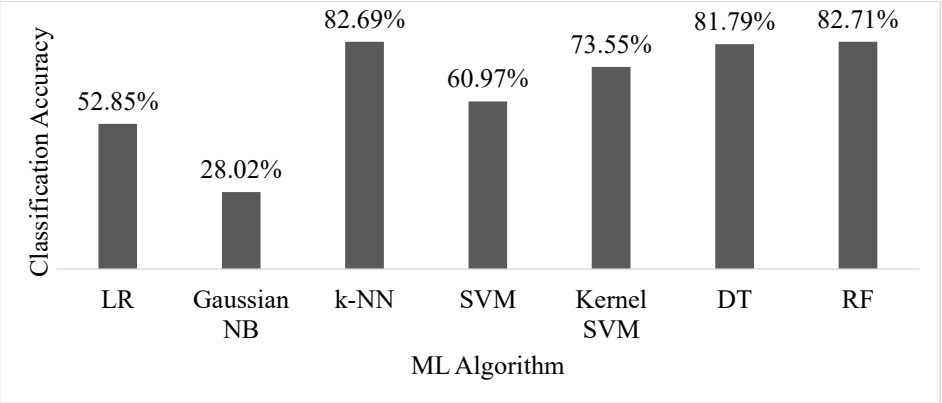


Figure 4.11
SCADA: Affected Component Overall Classification Accuracy (5-fold cross-validation)

of the fired alert compared to the binary classification explained in Section 4.4.1. Due to the fact that more classes are involved, the accuracy of the model accounts for a large number of false positives. This trade-off is further noted in Table 4.5 where the recall, precision, and F1-Score are listed for the different algorithms. LR and SVM experience the least recall of 39.83% and 59.82%. This is due to the non-linearity of the affected components classes. Therefore, the inability of these techniques to fit, whereas k-NN has a recall of 79.76%, DT has a recall of 74.31%, and NB has a recall of 79.76%, for the benign class.

Table 4.5
SCADA Results: Experiment 2 - Affected Component Classification (5-fold cross-validation)

Classification (Affected Component)	**Recall**	**Precision**	**F1-Score**
LR			
None	39.83%	31.38%	35.08%
Discrete Sensor 1	59.53%	37.16%	45.75%
Discrete Sensor 2	23.79%	60.17%	34.05%
Network	56.57%	53.66%	55.06%
Ultrasound Sensor	62.55%	70.09%	66.09%
Whole	69.57%	100%	81.98%
Weighted Average	52.85%	56.74%	56.74%
NB			
None	79.76%	38.48%	51.9%

Discrete Sensor 1	50.57%	35.71%	41.82%
Discrete Sensor 2	0%	0%	0%
Network	16.71%	21%	18.61%
Ultrasound Sensor	1.73%	98.89%	3.39%
Whole	100%	18.82%	31.67%
Weighted Average	28.02%	43.26%	43.26%
k-NN			
None	74.46%	78.39%	76.33%
Discrete Sensor 1	76.82%	74.17%	75.43%
Discrete Sensor 2	68.47%	63.09%	65.66%
Network	84.73%	87.45%	86.05%
Ultrasound Sensor	90.55%	90.59%	90.57%
Whole	99.13%	100%	99.56%
Weighted Average	82.69%	83%	83%
SVM			
None	59.82%	42.95%	50%
Discrete Sensor 1	84.69%	58.63%	69.25%
Discrete Sensor 2	23.79%	57.98%	33.69%
Network	54.27%	55.87%	55.05%
Ultrasound Sensor	66.83%	72.89%	69.72%
Whole	100%	100%	100%
Weighted Average	60.97%	62.05%	62.05%
Kernel SVM			
None	62.82%	67.27%	64.96%
Discrete Sensor 1	82.24%	61.62%	70.43%
Discrete Sensor 2	61.62%	51.93%	56.36%
Network	70.54%	79.86%	74.9%
Ultrasound Sensor	77.84%	85.76%	81.59%
Whole	99.57%	100%	99.78%
Weighted Average	73.55%	75.06%	75.06%
DT			
None	74.31%	75%	74.63%
Discrete Sensor 1	73.7%	74.46%	74.06%
Discrete Sensor 2	64.72%	64.11%	64.39%
Network	84.8%	85.29%	85.04%
Ultrasound Sensor	90.3%	89.4%	89.85%
Whole	99.89%	100%	99.95%
Weighted Average	81.79%	81.8%	81.8%

RF			
None	74.16%	78.24%	76.11%
Discrete Sensor 1	77.19%	73.68%	75.38%
Discrete Sensor 2	68.32%	64.32%	66.22%
Network	85.12%	86.43%	85.77%
Ultrasound Sensor	90.06%	91.18%	90.61%
Whole	99.89%	100%	99.95%
Weighted Average	82.71%	82.95%	82.95%

The results of this experiment confirm the diversity of scenarios that affect different components, yet the overlap between classes causes some to be harder to detect than others. For example, the detection of the "Discrete Sensor 2" class experiences a low recall of 23.79%, 0%, 68.47%, 23.79%, 61.62%, 64.72%, and 68.32% when using LR, NB, k-NN, SVM, Gaussian SVM, DT, and RF, respectively. On the other hand, the recall of the "Whole" class is high. The recall, in this case, is 69.57%, 100%, 99.13%, 100%, 99.57%, 99.89%, and 99.89% using LR, NB, k-NN, SVM, Gaussian SVM, DT, and RF, respectively.

Pointing out the affected component extends the reporting capability of the model. It allows a better identification of the problem, thus a quicker response. However, the optimal case would be to report the exact scenario. This will reduce the time spent to identify the problem and speed up the mitigation process. To this end, a third experiment is established.

4.4.3 Experiment 3: Scenarios Classification

In the third experiment, the goal is to further extend the system's ability to flag anomalies by identifying the specific scenarios that are considered suspicious. To perform this experiment, three operational trials are conducted. The experiments use multi-class classifiers and leverage their output probabilities to report the suspicious scenario.

4.4.3.1 One Scenario Classification

In the first trial, ML models are trained to classify different scenarios based on the multi-class classifier highest probability. Figure 4.12 shows the results of the different ML techniques. The highest accuracy reaches only 81.19%. For completeness and reproducibility, the full results tables (recall, precision, and F1-Score) are reported in Appendix C.

The results demonstrate either a high or a low recall for different classes. To analyse this, the scenarios are reviewed and the following conclusions are drawn:

(i) The scenarios in the dataset are co-related, meaning that they can overlap. This is a known problem in CI setup [232].

(ii) The models report multiple probable scenarios for each instance with a maximum of 4 probable ones.

Elaborating on the second conclusion, since the ML models can output the probability of the classification result, it is observed that each instance results in a maximum of 4 non-zero scenario probabilities. This means that each instance can belong to at most 4 scenarios. This is demonstrated in Table 4.6 where it lists the number, and percentage, of instances with multiple possible scenarios. Since the

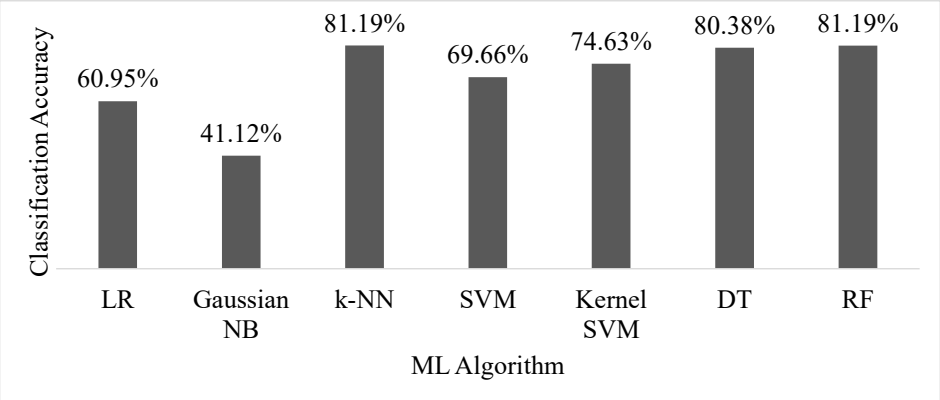


Figure 4.12
SCADA: Scenarios Overall Classification Accuracy, Single Scenario (5-fold cross-validation)

Table 4.6
SCADA: Distribution of Probabilistic Classification of Scenarios

Algorithm	Maximum number of probable scenarios per instance	Number of instance with **1** probable scenario	Number of instance with **2** probable scenarios	Number of instance with **3** probable scenarios	Number of instance with **4** probable scenarios
DT	1	3817	-	-	-
		100%	-	-	-
k-NN	3	2336	1221	260	-
		61.20%	31.99%	6.81%	-
RF	4	2144	1315	355	3
		56.17%	34.45%	9.3%	0.08%

maximum is 4 probable scenarios, the table shows the count of instances having 1, 2, 3, or 4 probable scenarios. For example, the second row shows that 61.2% of the instances are classified with only 1 probable class, 31.99% of the instances are classified with 2 probable scenarios and 6.81% with 3 probable scenarios. As a result, the following experiments leverage this to report two probable scenarios which reduces the uncertainty of this approach.

4.4.3.2 Two Scenarios Classification

In the second trial, the model reports two scenarios when an anomaly occurs instead of one, compared to the previous experiment. The two scenarios are the ones with the highest probabilities provided by the different classifiers. Based on the fact that the scenarios overlap and anomalies in CI are not mutually exclusive, an instance is correctly classified if it belongs to one of the two reported scenarios. In this case, multi-label classification would improve the detection accuracy. Figure 4.13 shows that this modification increases the overall detection accuracy to 95.60% and 95.49% when using RF and k-NN, receptively. A model with higher detection accuracy allows for a better action according to the reported scenarios. This alleviates the attack and reduces the overall response time needed.

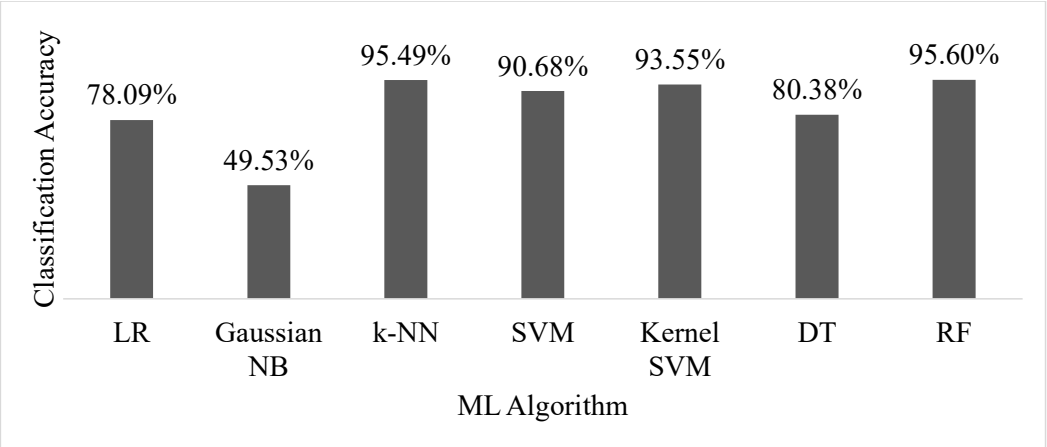


Figure 4.13
SCADA: Scenarios Overall Classification Accuracy, Two Probable Scenarios (5-fold cross-validation)

Table 4.7 provides a demonstration of instances that are correctly classified when one probable scenario versus two probable scenarios are reported. The provided demonstration is calculated in regard to the k-NN classification model. In the first row, "2 Floating Objects" scenario, 67 are misclassified as "Plastic Bag" sabotage. However, 60 of them can be correctly reported by considering the second probable

Table 4.7
SCADA: Co-relation of scenarios that are misclassified based on one probable scenario and correctly reported with the second probable one (Calculated based on k-NN experiment)

Scenario (X)	Instances count where X		Scenario (Y) The count of instances classified as Y while the correct is X						
	Is Not 1st Probable Scenario	**Is** 2nd Probable Scenario	2 Floating Objects	7 Floating Objects	Normal	Plastic Bag	Sensor Failure	Spoofing	Wrong Con.
2 Floating Objects	67	60	-	-	-	67	-	-	-
7 Floating Objects	5	5	-	-	-	-	-	-	5
Normal	113	78	-	-	-	1	96	-	16
Plastic Bag	107	91	49	-	3	-	35	20	-
Sensor Failure	242	184	-	-	99	73	-	9	61
Spoofing	54	32	-	-	-	44	10	-	-
Wrong Con.	134	96	-	7	31	8	88	-	-

scenario. Similarly, in Table 4.7 row 4, 107 instances of the “Plastic Bag” scenario are misclassified to be “2 Floating Objects” (49 instances), “Sensor Failure” (35), “Spoofing” (20 instances) and “Normal” (3 instances). 91 of these instances can be correctly reported with the consideration of the second probable scenario.

In this experiment, two probable scenarios are reported. This can be misleading when the first scenario is sufficient. This happens when an anomaly does not match multiple scenarios. In this case, reporting a second scenario adds unneeded complexity. Therefore, the third trial provides a confidence measure that allows better reporting, thus improving the situational handling response.

4.4.3.3 Scenarios Classification Using Confidence

In the third trial, a single scenario is solely reported unless its classification probability is less than a defined threshold. This threshold serves as the model confidence interval. Therefore, when the model classification probability falls below this threshold, two scenarios are reported. Two threshold values are used for this experiment; 75% and 85% and the results are shown in Figure 4.14.

When a 75% confidence interval is used, a single scenario is reported as long as its classification probability is greater than or equal to 0.75, otherwise, two scenarios are reported. The overall detection accuracy reaches 91.57%. The overall classification accuracy rises when using an 85% confidence interval. This is demonstrated in Figure 4.14. The overall detection accuracy reaches a maximum of 95.49% using k-NN.

Some conclusions can be drawn from Figure 4.13 and Figure 4.14. DT accuracy remains the same in all experiments due to the DT output which reports a single scenario for all the instances based on how the decision branches are formed. Table 4.6 shows that the DT model outputs a single scenario for 100% of the instances. Therefore, it is not possible to report two probable scenarios. The accuracy reaches 95.39%, hence reducing the uncertainty and allowing a fast alleviation of the attack.

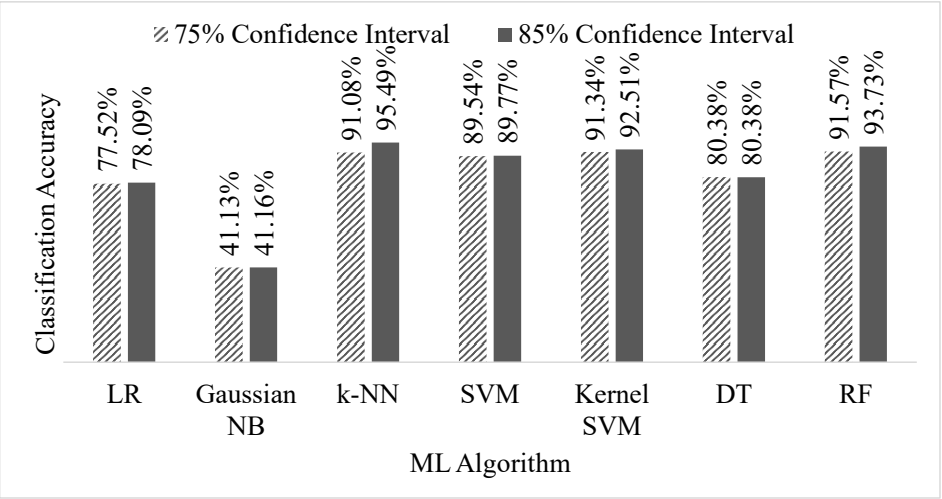


Figure 4.14
SCADA: Scenarios Overall Accuracy Classification, One or Two Scenario(s) Based on 75% and 85% Confidence Intervals (5-fold cross-validation)

In the next section, the same ML techniques are used to evaluate and analyse the performance of detecting anomalies in IoT networks. A novel MQTT-based dataset is generated and used for evaluation.

4.5 MQTT IDS Dataset Generation

IoT devices have been used extensively in the past decade and is estimated to reach 25.1 billion devices in 2025 [233]. IoT networks are utilised for different purposes [149] that include, but are not limited to, smart cities [234], farming [235], supply chain [236], and healthcare [237]. One of the distinguishable protocols for machine-to-machine IoT communication is MQTT [238, 212].

Harsha *et al.* [239] survey the protocols used in IoT networks with a focus on MQTT protocol. The authors' work identifies the various MQTT associated security risks, which highlights the need for special-purpose IDS. Their work shows that there are **53396** publicly available and accessible MQTT devices [239]. Dinculeană and Cheng [240] further analyse the MQTT security vulnerabilities. Their work concludes that there is a need for robust detection techniques for MQTT attacks.

Due to the lack of datasets that comprise IoT traffic, there is a pressing need to generate up-to-date IoT datasets for IDS usage. Based on the analysis and to the best of the author's knowledge, there are no available IDS datasets that contain MQTT traffic (benign or malicious).

Furthermore, with the increased dependence on IoT and the inadequacy of general-purpose IDS to fit IoT needs - there is a need to build IoT IDS. Nonetheless, this requires the availability of datasets to process, train, and evaluate classification models. In this section, a generated MQTT-based IDS dataset is presented. This is the first dataset to simulate an MQTT-based network comprising benign and malicious traffic (representing generic and MQTT-specific cyber attacks).

4.5.1 MQTT-IoT-IDS2020

The "MQTT-IoT-IDS2020" dataset is generated using a simulated MQTT network architecture to reflect IoT network communication. The network comprises twelve sensors, a broker, a simulated camera, and an attacker. Five scenarios are recorded which cover normal operation and four attack scenarios. The attacker performs the four attacks and each is recorded independently.

The attack scenarios aim to cover both MQTT-based attacks and generic attacks that are known to target IoT networks. According to Dietz *et al.*, the life cycle of an IoT network attack is composed of 7 stages. The first stage is "Scan", followed by "Brute-force" [241]. Network scanning is the first tier in the five-tier IoT threat model that is outlined by Hafeez *et al.* [242]. Therefore, the attack scenarios are:

- Aggressive scan (Scan_A)
- UDP scan (Scan_sU)
- Sparta SSH brute-force (Sparta)
- MQTT brute-force attack (MQTT_BF)

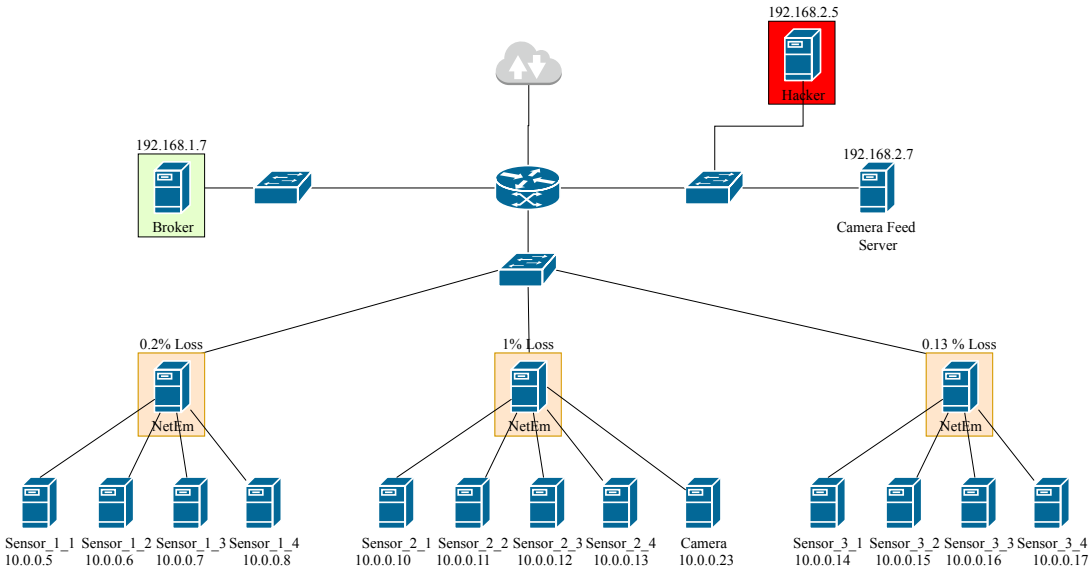


Figure 4.15
MQTT-IoT-IDS2020: Network Architecture

Figure 4.15 shows the network components. The broker is responsible for delivering the messages as it receives them from all subscribers and routing them to their destinations. The 12 sensors are divided into three subnetowrks to simulate three different components. During normal operation, all 12 sensors send randomised messages using the “Publish” MQTT command. The length of the messages varies between sensors to simulate different usage scenarios and the content is randomly generated. In order to simulate a camera feed, VLC media player is used to send a continuous UDP stream to the camera feed server. The network traffic is captured using the router’s Ethernet ports. Furthermore, the network drops packets to simulate real-life situations. The dropping rates are 0.2%, 1%, and 0.13%, which are based on the average acceptable loss rates [243]. The benign traffic (normal traffic) is first recorded. Then the operation continues as background traffic during the process of recording different attacks. As shown in Figure 4.15, the attacker IP address is “192.168.2.5”, which is used later in labelling the dataset instances.

Tcpdump is used to capture the network traffic. The following tools are used as follows:

- Virtual machines are used to simulate the network devices.
- Nmap is used for the scanning attacks.
- MQTT-PWN [244] is used for the MQTT brute-force attack.

The OS specification is as follows:

- Sensors: Tiny Core Linux
- Camera and camera feed server: Ubuntu
- Hacker: Kali Linux

For each scenario, a raw PCAP file is saved. The five files, namely; normal.pcap, sparta.pcap, scan_A.pcap, mqtt_bruteforce.pcap and scan_sU.pcap, are processed to extract three abstraction levels of features. The feature levels are: (i) Packet features, (ii) Unidirectional flow features, and (iii) Bidirectional flow features [245]. Flow based features represent the communication between two nodes in the network (for example, the average time between packets and the number of packets in a flow). The features are saved in CSV files that are suited for ML usage. The raw PCAP files are made available with open-access privileges [16] for further analysis of MQTT network communication and the associated attacks. It is important to note that the three feature abstraction levels are used exclusively as discussed in Section 4.6.

Table 4.8 summarises the features extracted from the raw PCAP files. The fourth column shows the packet-based features. Column five shows unidirectional flow features, and finally, column six shows bidirectional flow features. For the bidirectional flows, some features (pointed as *) have two values— one for the forward flow and one for the backward flow. The two features are recorded and distinguished by a prefix "fwd_" for forward and "bwd_" for backward. The distribution of instances is listed in Table 4.9.

Table 4.8
MQTT-IoT-IDS2020: Feature List and Description

Feature	Variable	Data Type	Packet	Uni-flow	Bi-flow
Source IP Address	ip_src	Text	✓	✓	✓
Destination IP Address	ip_dest	Text	✓	✓	✓
Last layer protocol	protocol	Text	✓		
Time to live	ttl	Integer	✓		
Packet Length	ip_len	Integer	✓		
Don't fragment IP flag	ip_flag_df	Binary	✓		
More fragments IP flag	ip_flag_mf	Binary	✓		
Reserved IP flag	ip_flag_rb	Binary	✓		
Source Port	prt_src	Integer	✓	✓	✓
Destination Port	prt_dst	Integer	✓	✓	✓
Transport Layer protocol (TCP/UDP)	proto	Integer		✓	✓
Reserved TCP flag	tcp_flag_res	Binary	✓		
Nonce sum TCP flag	tcp_flag_ns	Binary	✓		
Congestion Window Reduced TCP flag	tcp_flag_cwr	Binary	✓		
ECN Echo TCP flag	tcp_flag_ecn	Binary	✓		
Urgent TCP flag	tcp_flag_urg	Binary	✓		
Acknowledgement TCP flag	tcp_flag_ack	Binary	✓		
Push TCP flag	tcp_flag_push	Binary	✓		
Reset TCP flag	tcp_flag_reset	Binary	✓		
Synchronization TCP flag	tcp_flag_syn	Binary	✓		
Finish TCP flag	tcp_flag_fin	Binary	✓		
Number of Packets in the flow	num_pkts	Integer		✓	*
Average inter arrival time	mean_iat	Decimal		✓	*
Standard deviation of inter arrival time	std_iat	Decimal		✓	*
Minimum inter arrival time	min_iat	Decimal		✓	*
Maximum inter arrival time	max_iat	Decimal		✓	*
Number of bytes	num_bytes	Integer		✓	*
Number of push flag	num_psh_flags	Integer		✓	*
Number of reset flag	num_rst_flags	Integer		✓	*
Number of urgent flag	num_urg_flags	Integer		✓	*

Table 4.8 continued					
Feature	**Variable**	**Data Type**	**Packet**	**Uni-flow**	**Bi-flow**
Average packet length	mean_pkt_len	Decimal		✓	*
Standard deviation packet length	std_pkt_len	Decimal		✓	*
Minimum packet length	min_pkt_len	Decimal		✓	*
Maximum packet length	max_pkt_len	Decimal		✓	*
MQTT message type	mqtt_messagetype	Integer	✓		
mqtt_messagelength MQTT message length		Binary	✓		
User Name MQTT Flag	mqtt_flag_uname	Binary	✓		
Password MQTT flag	mqtt_flag_passwd	Binary	✓		
Will retain MQTT flag	mqtt_flag_retain	Binary	✓		
Will QoS MQTT flag	mqtt_flag_qos	Integer	✓		
Will flag MQTT flag	mqtt_flag_willflag	Binary	✓		
Clean MQTT flag	mqtt_flag_clean	Binary	✓		
Reserved MQTT flag	mqtt_flag_reserved	Binary	✓		
Is Attack	is_attack	Binary (1 if the instance is attack, 0 otherwise)	✓	✓	✓
* represented as two features in the biflow features file					

Table 4.9
MQTT-IoT-IDS2020: Instances Distribution

File Name	**PCAP file size**	**Number of Packets**		**Number of Uni-flow Instances**		**Number of Uni-flow Instances**	
		Benign	**Attack**	**Benign**	**Attack**	**Benign**	**Attack**
normal	192.5 MB	1056230 (3.42%)	0	171836 (59.01%)	0	86008 (54.78%)	0
scan_A (aggressive)	16.2 MB	70768	40624 (0.13%)	11560	39797 (13.67%)	5786	19907 (12.68%)
scan_sU (UDP)	41.3 MB	210819	22436 (0.07%)	34409	22436 (7.71%)	17230	22434 (14.29%)
sparta	3.4 GB	947177	19728943 (63.93%)	154175	28232 (9.7%)	77202	14116 (8.99%)
mqtt_bf	9.6 MB	32164	10013142 (32.45%)	4205	28874 (9.92%)	2152	14544 (9.26%)

4.6 MQTT Experiments and Results

In order to assess the effectiveness of different ML techniques on the MQTT-IoT-IDS2020 dataset, the six ML algorithms discussed earlier are utilised; LR, Gaussian NB, k-NN, SVM, DT and RF. The following features are excluded to ensure there is no influence of identifiable data: source and destination IP addresses, protocol, and MQTT flags. The data is split into 80% and 20% for training and testing, respectively, and five-fold cross-validation [231] is used to evaluate each experiment.

The overall accuracy is used for evaluation in addition to the precision, recall, and F1-Score of each class. Finally, the weighted average (W. AVG.) of precision, recall, and F1 score is calculated to further analyse the IDS performance.

Table 4.10 presents the overall accuracy of each of the ML techniques with each of the feature levels; packet, unidirectional and bidirectional. By observing Table 4.10, the overall accuracy increases when flow-based features are used for all algorithms except NB. This rise in accuracy can be seen in Figure 4.16 and is reasoned by the fact that flow-based features better discriminate benign and MQTT-based attacks.

Table 4.10
MQTT-IoT-IDS2020: Overall Detection Accuracy

	Features		
	Packet	**Unidirectional**	**Bidirectional**
LR	78.87%	98.23%	99.44%
k-NN	69.13%	99.68%	99.9%
DT	88.55%	99.96%	99.95%
RF	65.39%	99.98%	99.97%
SVM (RBF Kernel)	77.4%	97.96%	96.61%
NB	81.15%	78%	97.55%
SVM (Linear Kernel)	66.69%	82.6%	98.5%

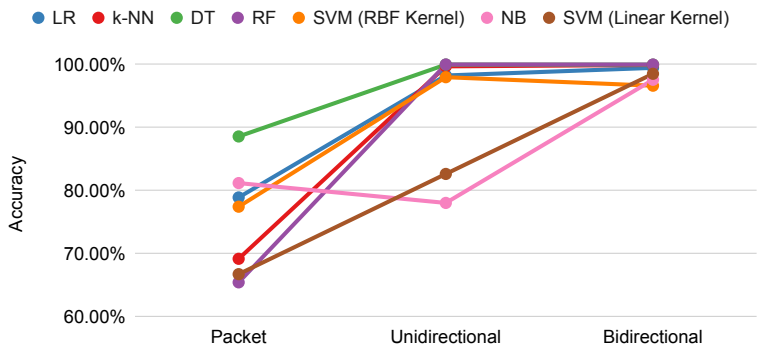


Figure 4.16
MQTT-IoT-IDS2020: Overall Detection Accuracy Trend using Different ML Techniques

Furthermore, RF in Table 4.10 demonstrate the highest overall accuracy when using flow-based features. The accuracy reached 99.96% when using unidirectional flow features and 99.97% when using bidirectional ones. This 0.01% difference between uni and bidirectional flow features is insignificant due to the fact that these results are the average of 5-fold cross validation. However, the results demonstrate the effectiveness of RF in this case, which is reasoned by the techniques ability to handle multiple features without overfitting, and its appropriateness for multiclass problems. This is further observed in Table 4.12 where the recall, precision, and F1-Score of all the five classes reach over 99.9%.

Table 4.11
MQTT-IoT-IDS2020 Results: LR - k-NN - DT (5-fold cross-validation)

	LR								
	Recall			**Precision**			**F1-Score**		
	Packet	**Uni**	**Bi**	**Packet**	**Uni**	**Bi**	**Packet**	**Uni**	**Bi**
Benign	0%	100%	99.02%	0%	93.33%	98.95%	0%	96.55%	98.99%
Scan_A	86.45%	70.87%	97.25%	98.39%	98.39%	97.21%	92.03%	82.39%	97.2%
Scan_sU	98.21%	98.03%	98.48%	99.34%	95.76%	100%	98.77%	96.88%	99.23%
Sparta	100%	100%	100%	98.22%	100%	100%	99.1%	100%	100%
MQTT_BF	100%	99.25%	99.58%	51.75%	99.82%	99.41%	68.2%	99.53%	99.5%
W. AVG.	78.87%	98.23%	99.44%	70.4%	98.32%	99.44%	72.97%	98.14%	99.44%

	k-NN								
	Recall			**Precision**			**F1-Score**		
	Packet	**Uni**	**Bi**	**Packet**	**Uni**	**Bi**	**Packet**	**Uni**	**Bi**
Benign	17.43%	99.69%	99.95%	17.42%	98.85%	99.59%	17.43%	99.27%	99.77%
Scan_A	99.99%	99.97%	100%	99.99%	99.85%	99.9%	99.99%	99.91%	99.95%
Scan_sU	99.99%	99.96%	100%	99.99%	99.96%	100%	99.99%	99.96%	100%
Sparta	100%	100%	100%	100%	100%	100%	100%	100%	100%
MQTT_BF	25.84%	99.3%	99.75%	25.85%	99.82%	99.97%	25.84%	99.56%	99.86%
W. AVG.	69.13%	99.68%	99.9%	69.13%	99.68%	99.9%	69.13%	99.68%	99.9%

Table 4.12
MQTT-IoT-IDS2020 Results: DT - RF - SVM - NB (5-fold cross-validation)

	DT								
	Recall			Precision			F1-Score		
	Packet	Uni	Bi	Packet	Uni	Bi	Packet	Uni	Bi
Benign	69.29%	99.92%	99.88%	69.39%	99.92%	99.91%	69.34%	99.92%	99.9%
Scan_A	100%	100%	100%	99.98%	99.95%	99.9%	99.99%	99.97%	99.95%
Scan_sU	99.98%	99.91%	100%	100%	100%	100%	99.99%	99.96%	100%
Sparta	100%	100%	100%	100%	100%	100%	100%	100%	100%
MQTT_BF	72.56%	99.95%	99.93%	72.47%	99.95%	99.93%	72.51%	99.95%	99.93%
W. AVG.	88.55%	99.96%	99.95%	88.55%	99.96%	99.95%	88.54%	99.96%	99.95%

	RF								
	Recall			Precision			F1-Score		
	Packet	Uni	Bi	Packet	Uni	Bi	Packet	Uni	Bi
Benign	9.34%	99.96%	99.93%	8.99%	99.94%	99.95%	9.16%	99.95%	99.94%
Scan_A	100%	100%	100%	99.98%	99.95%	99.95%	99.99%	99.97%	99.98%
Scan_sU	99.98%	99.91%	99.96%	99.99%	100%	100%	99.99%	99.96%	99.98%
Sparta	100%	100%	100%	100%	100%	100%	100%	100%	100%
MQTT_BF	15.15%	99.96%	99.97%	15.69%	99.98%	99.96%	15.42%	99.97%	99.97%
W. AVG.	65.39%	99.98%	99.97%	65.44%	99.98%	99.97%	65.41%	99.98%	99.97%

	SVM (RBF Kernel)								
	Recall			Precision			F1-Score		
	Packet	Uni	Bi	Packet	Uni	Bi	Packet	Uni	Bi
Benign	30.23%	100%	100%	28.13%	92.67%	87.13%	28.8%	96.19%	93.12%
Scan_A	83.8%	70.16%	42.13%	99.99%	96.18%	99.88%	91.18%	81.13%	59.22%
Scan_sU	92.33%	99.96%	100%	99.74%	93.01%	94.34%	95.89%	96.36%	97.09%
Sparta	100%	100%	100%	91.17%	100%	100%	95.38%	100%	100%
MQTT_BF	72.42%	98.44%	98.3%	53.56%	100%	100%	59.53%	99.22%	99.14%
W. AVG.	77.4%	97.96%	96.61%	74.35%	98.05%	97.02%	74.89%	97.87%	96.15%

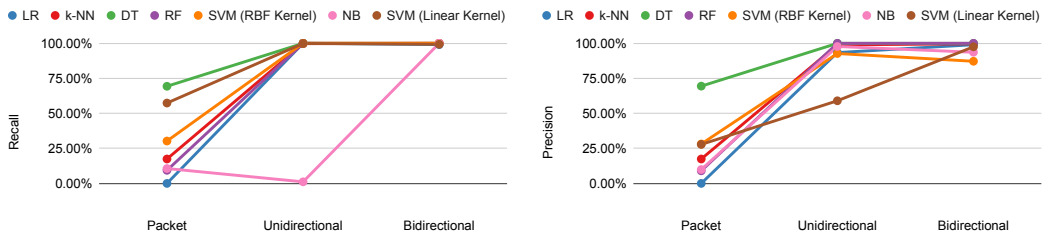
	SVM (Linear Kernel)								
	Recall			Precision			F1-Score		
	Packet	Uni	Bi	Packet	Uni	Bi	Packet	Uni	Bi
Benign	57.34%	99.84%	99.26%	27.8%	58.95%	97.45%	37.38%	73.82%	98.32%
Scan_A	83.28%	68.23%	84.1%	70.42%	70.35%	93.44%	69.7%	67.5%	87.01%
Scan_sU	78.13%	60.31%	97.76%	75.8%	70.71%	93.77%	76.92%	61.91%	95.27%
Sparta	87.64%	60.37%	99.99%	97.62%	99.94%	100%	89.89%	74.61%	99.99%
MQTT_BF	24.89%	97.79%	98.71%	43.3%	99.89%	99.55%	20.84%	98.83%	99.13%
W. AVG.	66.69%	82.6%	98.5%	65.42%	88.9%	98.66%	60.4%	82.42%	98.46%

	NB								
	Recall			Precision			F1-Score		
	Packet	Uni	Bi	Packet	Uni	Bi	Packet	Uni	Bi
Benign	10.62%	1.13%	99.96%	9.9%	97.68%	93.56%	10.25%	2.24%	96.65%
Scan_A	100%	99.25%	66.41%	99.23%	18.28%	100%	99.61%	30.88%	79.81%
Scan_sU	99.52%	97.76%	100%	100%	98.79%	98.52%	99.76%	98.27%	99.25%
Sparta	99.84%	100%	100%	100%	100%	100%	99.92%	100%	100%
MQTT_BF	90.27%	97.78%	100%	53.15%	100%	97.05%	65.84%	98.88%	98.5%
W. AVG.	81.15%	78%	97.55%	73.29%	95.43%	98.37%	75.99%	75.26%	97.77%

The importance of flow-based features is analysed by observing the separate classes metrics for each of the algorithms in Table 4.11 and Table 4.12 which outline the precision, recall, and F1-Score for each of the classifiers. In agreement with Table 4.10, the flow-based features usage improves the performance.

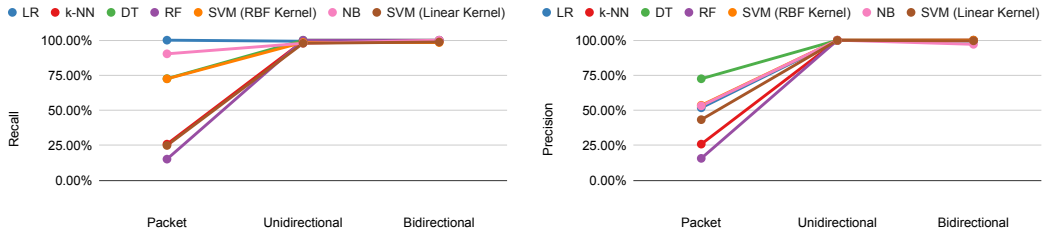
The two classes, for which performance significantly improves using flow-based features, are the benign class and the MQTT-BF attack class. In IoT networks, benign operation traffic is uncomplicated compared to general-purpose networks. This is based on the IoT network usage and requirements. Therefore, when an attacker initiates a general-purpose network-based attack, like scanning for example, it is distinctive. However, the challenge lies in MQTT-based attacks as they rely on the known MQTT communication commands (i.e., publish, subscribe, etc). Thus, packet-based features fail to discriminate benign from MQTT_BF attack across all the ML techniques used.

It is observed that the NB classifier experiences a distinguishably low performance with the benign class recall. This is because both packet-based features and unidirectional flow features are non-discriminative for benign traffic when using the conditional probabilistic approach that this classifier relies on. Therefore, the conditional probability of normal operation and MQTT_BF attack, given these features are not distinguishable. This behaviour is observed in the performance trend charts. Figure 4.17 shows the rise in the recall and precision of benign traffic, Similarly, Figure 4.18 plots the recall and precision of the MQTT_BF attack and finally, this rise is reflected in the overall weighted average recall and precision in Figure 4.19.



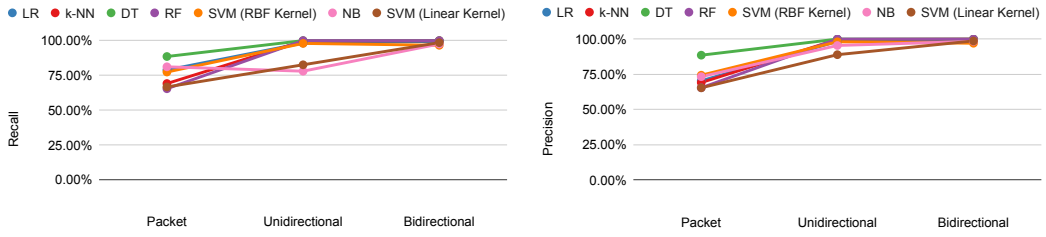
(a) Recall **(b)** Precision

Figure 4.17
MQTT-IoT-IDS2020: Benign Class Performance Trends



(a) Recall **(b)** Precision

Figure 4.18
MQTT-IoT-IDS2020: MQTT_BF Class Performance Trends



(a) Recall **(b)** Precision

Figure 4.19
MQTT-IoT-IDS2020: Weighted Average Trends

4.7 Summary

This chapter addresses the lack of special-purpose IDS and proposes solutions to some of the different challenges of building IDS for IoT and CI networks. Using the SCADA dataset, an anomaly detection IDS is built. The dataset covers 14 different real-world scenarios that include normal system behaviour, hardware failure, sabotage, and cyber attacks. Six ML techniques are used for evaluation and three experiments are conducted. The experiments vary based on the level of information reported to the operator. The first experiment performs a binary classification (benign/anomaly). While instances are being detected as either anomaly or not, the type of anomaly is unknown, thus delaying any corrective actions. The second experiment reports the affected component of the occurring anomaly, improving the reporting capability of the model. Finally, the third experiment, which is the most reliable, reports the scenario. This helps in taking the required corrective actions and speed up the mitigation process. The code is

available on GitHub at https://github.com/AbertayMachineLearningGroup/machine-learning-SIEM-water-infrastructure.

The overall evaluation shows that k-NN, DT, and RF outperform NB, SVM, and LR. Moreover, k-NN results demonstrated the highest accuracy amongst all algorithms in the three experiments. The accuracy reached 94.12% for the binary classification and 95.49% for the scenarios classification. Since the scenarios overlap (i.e., are co-related), a confidence level is used to provide the operator with the most probable scenario and two probable scenarios when the confidence is low.

Using MQTT as an IoT case study, an IoT network is simulated and a novel dataset is generated. The dataset covers normal operation, general networking attacks, and MQTT-based attacks. The dataset is initially saved as raw PCAP files, then features are extracted. Three levels of features are used, packet, unidirectional, and bidirectional features. Each feature level is used independently and six different ML techniques are used for attack classification. The dataset is available at https://ieee-dataport.org/open-access/mqtt-iot-ids2020-mqtt-internet-things-intrusion-detection-dataset .

The experiments demonstrate that networking attacks are efficiently discriminated from normal operation. This is because, in IoT setup, the normal operation patterns are simple and the generic networking attacks patterns are distinctive. However, protocol-specific attacks (MQTT-attacks) are harder and more complicated to classify due to their overlap with normal operation.

To overcome this, flow-based features are used and the experiments show that they are better suited to discriminate between normal and MQTT-based attacks due to their similar characteristics. Using the MQTT-IoT-IDS2020 dataset, the weighted average recall rose from $\sim$75.31% for packet-based features to $\sim$93.77% and $\sim$98.85% for unidirectional and bidirectional flow features, respectively. The weighted average precision rose from $\sim$72.37% for packet-based features to $\sim$97.19% and $\sim$99.04%

for unidirectional and bidirectional flow features. The experiments emphasise the special challenges faced by IoT IDS, based on their custom communication patterns. The challenges are demonstrated through the difficulty to differentiate MQTT-based attacks from normal operations. The code is available on GitHub at https://github.com/AbertayMachineLearningGroup/MQTT_ML.

Chapter 5

IDS using Limited-Size Data

5.1 Problem Statement

Special-purpose IDS, as well as general-purpose ones, need large datasets to train IDS models [23]. Datasets are often depicted as the bottleneck for developing robust ML models, including ML-based IDS, due to the following reasons [246]: (i) Gathering large realistic datasets is a complex task that requires a lot of processing time. (ii) Training with small datasets exposes the ML model to overfitting problems.

Acquiring large volumes of training data poses a particular problem with IDS that defers their advancement for two main reasons. (i) The need for continuous generation of datasets to cope with zero-day and emerging attacks. This is impractical in real-time. and (ii) the long interim time between a new cyber attack being detected and building a corresponding dataset that contains representative instances of this new attack. By the time a large dataset is generated to mimic a new cyber attack and the retraining process takes place, newer cyber attacks are detected and more data is needed as cyber attacks emerge at an exponential rate [8].

This dataset dependency problem can be defined as resolving the directly proportional relation between the complexity of a problem, the size of a required

model, and the amount of data needed as shown in the following expression: "$Size(Model) \propto$ **Size(Data)** $\propto Complexity(Problem)$" [247]. This chapter focuses on resolving this relation in regard to the size of the data, which requires a shift in the development process of ML models.

Formally, this chapter proposes a new approach for building IDS. The approach relies on One-Shot learning paradigm which enables training using limited size datasets, thus, alleviating the need to gather large datasets. To this end, a Siamese network model is proposed and trained to differentiate between classes based on pair similarities rather than specific class features. Learning from similarities requires less data for training and provides the ability to introduce new cyber attacks post-training (i.e. zero-day attacks).

The proposed Siamese network is evaluated for three usage scenarios. The first scenario evaluates the validity of similarity-based learning for IDS usage. This is performed by assessing the classification accuracy using limited data for training. The second usage scenario evaluates the ability of the Siamese network model as a One-Shot learning model by introducing new attack classes that are not used during training. Finally, the third scenario evaluates the effectiveness of similarity-based learning to detect unknown zero-day attacks.

Four datasets are used. The first dataset is the SCADA dataset, which represents CI setup where dataset availability is limited and hard to gather. Then, the model is generalised for general-purpose IDS datasets. The recent CICIDS2017 dataset is used alongside the most used datasets for IDS evaluation; KDD Cup'99, and NSL-KDD.

5.2 Background

5.2.1 Learning from Limited-Size Datasets

Li *et. al* [11] discuss the large dataset requirements and the difficulty of obtaining such datasets. Besides the size of the dataset, current ML approaches require an extensive amount of time to train a single model. Therefore, researchers propose approaches to handle this time and dataset size limitations.

Online learning focuses on reducing the computation time needed to train a model. This is done by continuously updating the model weights (i.e. tuning) as data becomes available. This learning paradigm assumes that data becomes available over time and does not require the dataset to be fully available prior to training. [248]. However, caution must be taken when utilising online learning because models can shift to undesirable states over time as training continues [249]. Moreover, online learning is not suitable to learn from small datasets, nor detect unknown attacks.

Prior research suggested "*Transfer Learning*" to overcome the need for large datasets [250, 251]. The premise of transfer learning to solve a target problem T (where data availability is limited), is to create a model M for a similar problem T', where large amount of data is available. The model M is then transferred to the initial problem T and retrained on the limited dataset. The rationale for transfer learning is that the initial training on T', yields training weights that discover useful features for the problem domain and hence suitable for problem T. Therefore, after retraining, the model learns and generalises faster on small datasets [252]. This is a common approach in the image processing domain where, for example, models are trained using standard large datasets, such as, MNIST and ImageNet [253, 254, 255].

Despite the fact that transfer learning proposes a viable solution, it does not eliminate the need for an original large dataset and raises a number of additional challenges; (i) Finding a suitable pretrained model *"What to transfer?"* [250],

(ii) Deciding the appropriate tuning of the pretrained model to fit in the new domain *"How and when to transfer?"* [250] and (iii) Transfer could reduce the learning performance of the target domain, known as "Negative Transfer" [250, 256].

To overcome the need to build new datasets for detecting unknown attacks, X. Sun *et al.* [257] proposed a Bayesian probabilistic model to detect zero-day attack paths. The authors visualised attacks in a graph-like structure and introduced a prototype to identify them. Their results show the applicability of the proposed approach, however, the model is limited to the duration contained in the analyses period and restricted by the interaction with system calls.

Unlike the formerly discussed approaches and traditional ML techniques, One-Shot learning requires one or a few samples from each class to use during training, therefore, overcoming the need for large datasets. It also provides the ability of classifying classes that are not included in the training process. Chopra *et al.* explain this by mentioning that "traditional techniques are intrinsically limited to a small number of categories" [258].

In this chapter, the model proposed is designed based on the One-Shot learning paradigm using "Siamese Network". Siamese networks are trained to learn pair similarities rather than features to discriminate each class. Accordingly, given a small dataset, generating pairs of similar and dissimilar samples will instantly increase the size of the dataset, resulting in an average size dataset suitable for training. Moreover, since the network is trained to detect similarities, adding new cyber attack classes will be possible in real-time without the need for retraining.

5.2.2 One-Shot Learning

Fei-Fei *et al.* [259] were the first to introduce One-Shot learning. One-Shot learning is inspired by human being learning and generalisation capability, and focuses on learning new classes using only one - or a few - samples. One-Shot learning has

been used in various domains with the most prominent one being image and video processing [260, 261, 262]. It has also been introduced in other domains, such as, robotics [263], language processing [264, 265] and drug discovery [266].

5.2.3 Siamese Network

Siamese networks are widely used in the literature for One-Shot learning. Siamese networks were first introduced by Bromley *et al.* [267] in the 90's and were initially used to solve the problem of hand-written signature matching. Subsequently, Siamese networks were adapted by other domains. Popular implementations of Siamese networks exist for image and video processing. Koch *et al.* [268, 269] present one of the principle implementations of Siamese networks which is employed by many researchers. Other widely used implementations include: Yao *et al.* [270] and Varior *et al.* [271].

Figure 5.1 represents the Siamese network architecture which is composed of two identical subnetworks that share weights. The two networks are referred to as "Twin networks" and share a common architecture, i.e., two identical networks. The weights of the twin networks are initialised with random weights and the twin networks outputs are passed to a similarity component, which is responsible for calculating the distance defining "how alike" the two inputs are. The output of the latter is a comparison based similarity value. The loss is then calculated and the weights are updated based on gradients to minimise the loss function. Gradient descent is an optimisation algorithm that searches for the local (or global) minimum of a function [272]. During ANN training process, weights are updated by repeatedly taking steps in the opposite direction of the gradient until reaching a local minimum. The steps are determined based on the value of the learning rate [272].

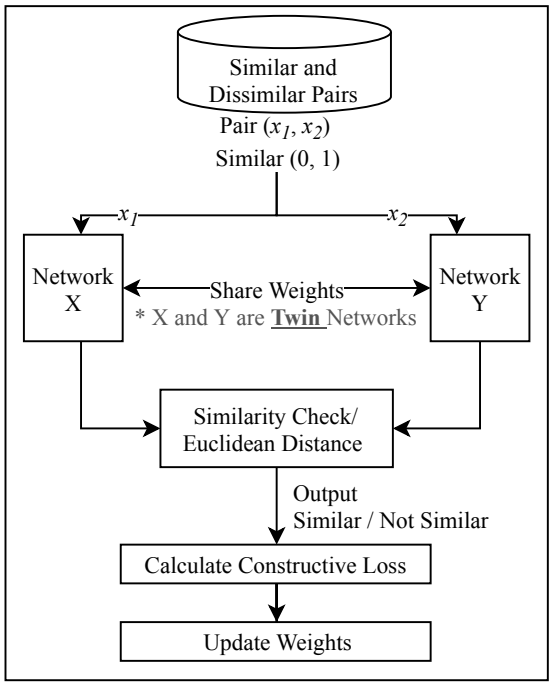


Figure 5.1
Siamese Network Architecture

Formally [268, 273], given a pair of inputs (x_1, x_2) and twin networks (X, Y), such that x_1 is the input of X and x_2 is the input of Y, the similarity between $f_1(x_1)$ and $f_2(x_2)$ can be computed using L2 norm (Euclidean distance) (Equation 5.1):

$$
\begin{aligned}
||d||_2 &= \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \\
||d||_2 &= \sqrt{\sum_{i=1}^{n}(f_1(x_1)_i - f_2(x_2)_i)^2}
\end{aligned}
\tag{5.1}
$$

such that f_1 and f_2 are the outputs of Networks X and Y respectively, and $f_1 \equiv f_2$ since X and Y are twin networks. Ultimately, the training goal is to minimise the overall loss l as defined in Equation 5.2; for each given batch i of input pairs $(x_1, x_2)_i$ and label vector y_i, where $y_i(x_1, x_2)_i = 1$ if x_1 and x_2 belong to the same class and 0 otherwise. λ represents l_2 regularisation parameter.

$$l(x_1, x_2)_i = \\ y(x_1, x_2)_i \log d_i + (1 - y(x_1, x_2)_i) \log(1 - d_i) + \lambda w^2 \tag{5.2}$$

This loss function is sensitive to outliers (i.e., dissimilar pairs with large distances), which disproportionately affect the gradient estimation. An alternative loss function is the constructive loss, proposed by Chopra, Hadsell and LeCun [258]. This is shown in Equation 5.3, where $m > 0$ is a margin. The constructive loss caps the contribution of dissimilar pairs if the distance is within a specified margin m [274], hence limiting the effect of large distances. In this work, the margin is set to $m = 1$ [274].

$$l(x_1, x_2)_i = y(x_1, x_2)_i \times (d_i)^2 \\ + (1 - y(x_1, x_2)_i) \times (max(m - d_i, 0))^2 \tag{5.3}$$

Batches of similar and dissimilar pairs are used to train the network. It is essential to note that an equal number of similar and dissimilar pairs are used in the training batches to avoid biases. After training, given any two pairs, the network is capable of calculating their degree of similarity, $d_i \in [0, 1]$; the lower the d_i, the closer the pair.

The choice of the twin networks architecture is domain specific and based on the application context. ANN, Convolutional Neural Network (CNN), and Long Short-Term Memory (LSTM) are commonly used architectures for establishing twin networks. CNN are well-suited for image processing whilst LSTM are routinely used with temporal data.

In this Chapter and for the purpose of the experiments, feedforward ANN are used as the building block of the twin networks as their structure is aligned with the structure and format of the data used in this Chapter for IDS purposes.

Similar to a single ANN, the Siamese network twin networks are trained in a back-propagation fashion. The twin networks are initialised with identical weights and during training, batches of similar and dissimilar pairs are used to calculate the loss, using the function given in Equation 5.3. The weights are then updated based on the learning rate, gradient descent, and optimisation function as explained in Section 5.2.4. Hyperparameter optimisation is performed to determine the model's set of optimal parameters. Hyperparameters were chosen based on consideration of: (a) random search [275], (b) recommendations by Lake *et al.* [276], who published their progress and findings on a 3-year project where Siamese networks were used for one-shot learning, Pang *et al.* [277], who published their Siamese network model that outperforms state-of-the-art in image processing domain, and (c) empirical analysis of ANN architecture. The details of the optimised architecture (i.e., the number of layers, neurons, etc.) are provided later in the Chapter.

5.2.4 Artificial Neural Networks

ANN are used as the building block of the Siamese network model in this chapter and the classification models the following chapters. ANN are inspired by how the biological brain works. The first ANN were proposed by McCulloch and Pitts [278] in 1943. Later in 1986, the back propagation paradigm was introduced by Rumelhart and McClelland [279].

ANN are composed of an input layer, zero or more hidden layers, and an output layer. Each layer is composed of one or more neurons. Each neuron has one or more input, and its computed output is passed onto the neurons in the following layer. Neurons in layer i are connected to the ones in layer j, $j = i + 1$. This connection between neurons is called weight and is represented as w_{ij}. During the training process, the input values are propagated forward, the error is calculated (based on the difference of the actual output and the expected one), then the error is propagated, and the

weights are updated accordingly. The value of a connection (i.e. weight) implies the significance of the input.

Formally, the output of a single neuron is calculated as shown in equation 5.4.

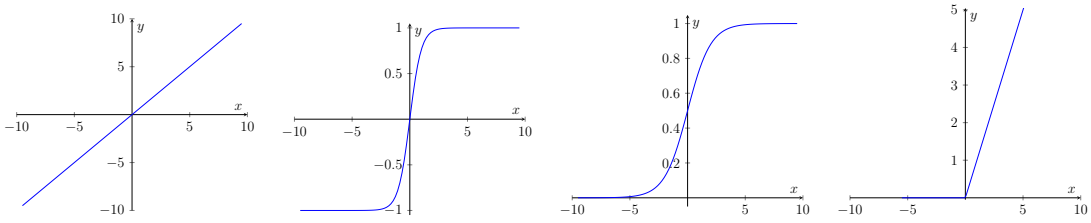
$$y_i = f((\sum_{i=0}^{n} x_i.w_i) + b) \tag{5.4}$$

Where n represents the number of inputs to this node, x_i is the i^{th} input value, w_i is the weight value, b is a bias value. Finally, f is the activation function, which squashes the output. This output can be the input to the next layer, or the final output of the network. Activation functions' main role, in the hidden layers, is to add non-linearity into the model [280]. In the final (output) layer, an activation function can be used to squash the output to the corresponding class labels, which is used to represent probabilities of the classification. Activation function can be, but not limited to, Linear, Tanh, Sigmoid, and Rectified Linear Unit (RELU) [280], shown in Figure 5.2.

The error E is calculated at the final layer using the difference between the expected output and the predicted output (which is, as mentioned, a result of propagating the input signal). Finally, the weights are updated based on Equation 5.5.

$$\boldsymbol{W}_{t+1} = \boldsymbol{W}_t - \eta \frac{dE}{d\boldsymbol{W}_t} \tag{5.5}$$

Where $\boldsymbol{W}_t$ is the old weight and $\boldsymbol{W}_{t+1}$ is the new weight. η is the learning rate to control the gradient decent steps.



(a) Linear **(b)** Tanh **(c)** Sigmoid **(d)** ReLU

Figure 5.2
ANN Activation Functions

The base of all optimisers is the gradient descent, as explained above. However, gradient decent encounters the problem of making big changes and could miss the minimum. As a result, stochastic gradient descent was introduced, where smaller subsets (random or batches) are used to calculate the gradient and more frequent updates are applied, then, momentum and acceleration were added [281]. One of the widely used optimisers is Adam (adaptive moment estimation) [282], where momentum is utilised by adding fractions of previous gradients.

The weight of a neuron is directly proportional to the significance of the node's input, which indicates the strength of the connection [283]. This is because the output of any neuron is calculated by multiplying the weights by the input values [284].

5.3 Datasets

Four datasets are used for the evaluation of the Siamese network model. The datasets cover the CI dataset introduced in Chapter 4 and three prominent general-purpose IDS datasets. The latter covers two benchmark IDS datasets, specifically, CICIDS2017 and NSL-KDD. Moreover, KDD Cup'99 is used in comparison to the NSL-KDD to demonstrate the effectiveness of having clean data when generating training pairs and also, when introducing new attacks to the trained model.

The four datasets are used to mimic the situations in which limited data is provided due to privacy and/or ethical concerns. The first dataset is discussed in detail in Section 4.3.

5.3.1 KDD Cup'99

The KDD Cup'99 [78] is considered the oldest benchmark dataset used in evaluating IDS. As outlined in Chapter 3, more than 60% of the research in the past years (2008 - 2020) was evaluated using this dataset. KDD Cup'99 comprises normal activity and 4 cyber attack classes.

The KDD Cup'99 dataset is relatively large, however, the dataset provider publishes a reduced subset of ∼10%. For the purposes of this experiment, only the smaller KDD Cup'99 10%, which covers all classes [285] is used to ensure the applicability of the proposed Siamese network to limited datasets. Table 5.1 shows the number of instances per class for the KDD Cup'99 dataset.

Table 5.1
KDD Cup'99 Classes and Corresponding Number of Instances

	Class	# of Instances
1	Normal	97278 (19.70%)
2	DoS	391458 (79.24%)
3	Probe	4107 (0.82%)
4	U2R	1128 (0.23%)
5	R2L	52 (0.01%)

5.3.2 NSL-KDD

The NSL-KDD [77] dataset is proposed by the Canadian Institute for Cybersecurity (CIC) to overcome the problems of the KDD Cup'99 dataset discussed by Tavallaee *et al.* [286]. Similar to KDD Cup'99, NSL-KDD covers 4 cyber attack classes and normal activity. The NSL-KDD is used for evaluation to observe the effect of enhancing and filtering a dataset on the similarity learning and performance. Table 5.2 shows the number of instances per class for the NSL-KDD dataset.

Table 5.2
NSL-KDD Classes and Corresponding Number of Instances

	Class	# of Instances
1	Normal	67343 (53.46%)
2	DoS	45927 (36.47%)
3	Probe	11656 (9.25%)
4	U2R	995 (0.78%)
5	R2L	52 (0.04%)

5.3.3 CICIDS2017

The CICIDS2017 dataset [62] is a recent dataset generated by the CIC. The dataset contains up-to-date real-life benign, insider and outsider attacks. Using the provided PCAP files, the bidirectional traffic flows are generated and labelled. Table 5.3 lists the attacks used in the experiments and the number of instances/flows for each. The dataset contains DoS attacks using different tools to initiate the attack, for example, HTTP Unbearable Load King (Hulk) and Slowloris [63].

Table 5.3
CICIDS Classes and Corresponding Number of Instances

	Class	# of Instances
1	Normal	248607 (90.50%)
2	DoS (Hulk)	14427 (5.25%)
3	DoS (Slowloris)	2840 (1.03%)
4	FTP Brute-force	5228 (1.9%)
5	SSH Brute-force	3627 (1.32%)

The NSL-KDD and KDD Cup'99 are provided in feature-like format, thus, they were preprocessed before being published. They have 42 features that are transformed to a total of 118 features after encoding the categorical features [287] . Finally, for the CICIDS2017, 31 bidirectional flow features are extracted.

It is essential to note that no feature engineering or selection is performed to ensure that the class excluded from training does not indirectly influence the feature sets in any way.

5.4 Siamese Network Usage Scenarios Overview

This section overviews the three different usage scenarios for Siamese networks. A conceptualisation of these scenarios is provided in Figure 5.3.

The first scenario is aimed to evaluate the suitability of similarity learning for cyber attack classification. In this scenario, the model is trained using a limited sample of

instances from K classes. A multi-class classification is performed, such that a new instance P is classified as one of the K classes.

The second scenario is the one based on One-Shot learning. The IDS is trained using instances from K classes. After training, labelled instances from a new class e are added. During the evaluation, a new instance P is classified as one of $K+1$ classes (i.e., the K classes that are used during training and an additional class e). The model is evaluated based on its efficiency to classify (i) a new attack class, and (ii) attack classes that are used during training.

Finally, the similarity-based training is used to detect zero-day attacks (i.e., cyber attacks that were never seen before by the model and no few labelled instances are available to fit in the second usage scenario). After training, the similarity measure is used to decide if a new instance belongs to one of the K known classes or an unknown class (i.e. zero-day attack).

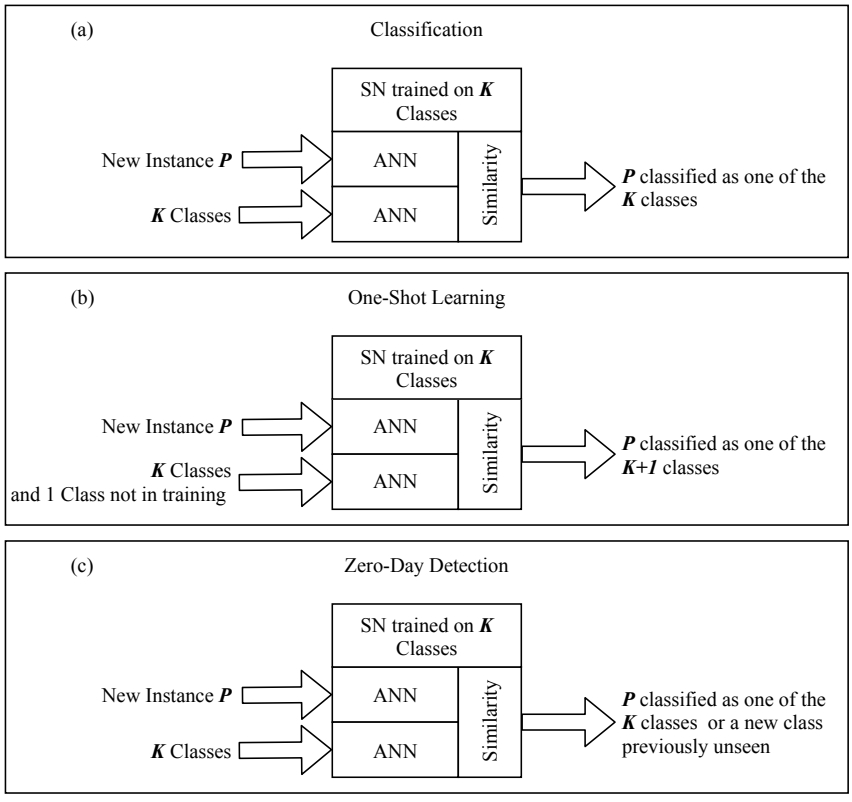


Figure 5.3
Siamese Network Usage Scenarios Overview

5.5 Scenario 1: Classification using Limited Data

In this section, the Siamese network model is utilised to classify instances from a group of known attack classes. The network is trained using pairs that can be obtained from a few samples of each class. This reduces the demand of collecting and annotating large datasets, and will validate the similarity-based learning paradigm for IDS development.

5.5.1 Methodology

Figure 5.4 visualises the process of building the classification intrusion detection model. The dataset is split into two halves, as shown in Figure 5.4-1. Collectively, the first half is used as the training pool of instances to generate similar and dissimilar pairs for training and validation sets (Figure 5.4-2). The second half is used as the evaluation pool of instances used to generate the testing pairs (Figure 5.4-3).

Since the Siamese network model relies on random pair generation, pairs are drawn out randomly from the pools of instances. The rationale for having pools of instances and randomly drawing out pairs, is to hinder any selection bias either during training (i.e., selecting similar and dissimilar pairs) or during evaluation of the new class (i.e. selecting the labelled instances that best represent this class). Furthermore, the uniqueness of the pairs - no duplicates - is ensured. It is important to note that the construction of similar and dissimilar pairs is an open research question in the literature [288].

For testing, each instance i is paired with one random instance from each class which is picked from the testing pool of instances producing N pairs. After the pairs are selected, the similarity is calculated for all pairs and the label (class) of instance i is decided based on the pair with the highest similarity (i.e., least distance).

In order to evaluate the trade-off between the number of labelled instances needed to represent each class and the classification accuracy, the pairing process is repeated

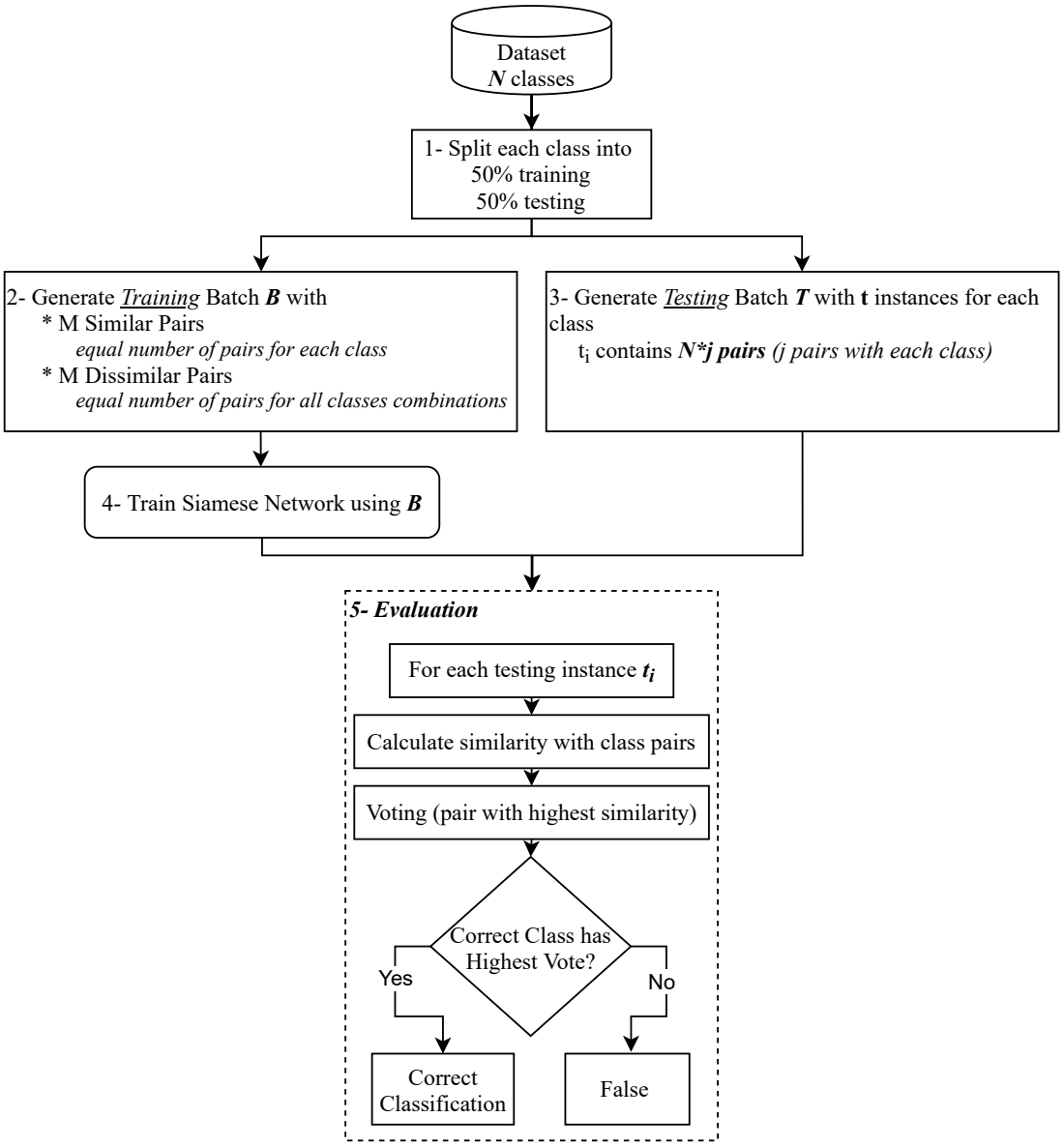


Figure 5.4
Siamese Network for Intrusion Detection (Classification)

j times for each instance i. A majority voting is used to deduce instance i label, where the class with the highest votes is selected as instance i label (Figure 5.4-5).

Algorithm 5.1 summarises the overall process of generating pairs, training, and testing the Siamese network model. Siamese twin networks architecture is determined by the number of input neurons being the number of features and one neuron as the output layer. The number of hidden layers and number of neurons in each layer is then determined; each hidden layer has a number of neurons that is reduced by a fraction

Algorithm 5.1 Siamese Network: Usage Scenario 1 Train and Test Algorithm

Input: Attacks Dataset
Output: Trained Siamese Network Evaluation
Ensure: $dataset = \{c_1, c_2, \ldots, c_n : n \geq 3\}$
1: $train\_batch\_size, test\_batch\_size \leftarrow 30,000$
2: $n\_epochs \leftarrow 2000$
3: $training = 50\%\ c_i\ \forall c_i \in dataset$
4: $testing = dataset - training$
5: $batch \leftarrow$ GETTRAININGBATCH($train\_batch\_size$)
6: Build Siamese Network with Random Weights
7: **for** $i = 0$ to $n\_iterations$ **do**
8: Update Siamese Network Weights based on $batch$
9: **end for**
10: EVALUATECLASSIFICATION(test_batch_size)

from the previous layer [289]. The tuning of the architecture is performed using ANN parameter optimisation. During the training phase, both training and validation loss curves are monitored to ensure that the network converges, while avoiding overfitting, using Dropout layers. The parameters (the number of hidden layers, number of neurons in each layer, η - learning rate -, number of epochs, etc) are chosen based on the optimal state of the model.

The Siamese network regularisation can be monitored using the loss behaviour. Regularisation is carried out on the onset of unstable behaviour during training. Figure 5.5 shows an unstable network performance state.

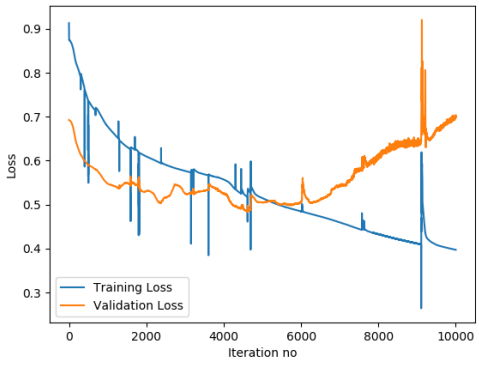


Figure 5.5
Siamese Network Loss Curve (Non-converging case)

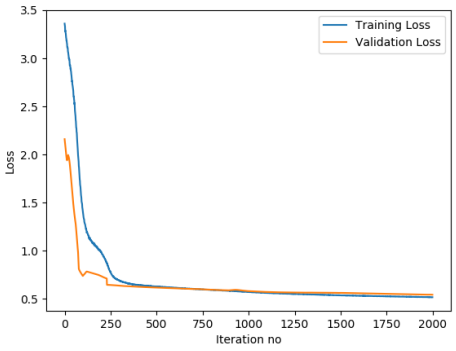


Figure 5.6
Siamese Network Loss Curve
(Converging case) - 1

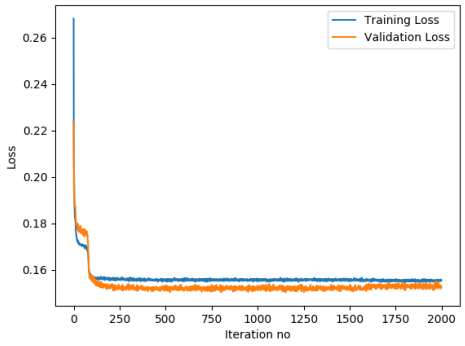


Figure 5.7
Siamese Network Loss Curve
(Converging case) - 2

As a result, the regularisation parameters of the network are reconsidered and dropout layers and kernel regularisation are added to obviate overfitting and ensure network convergence. This is distinctly observed in Figure 5.6 and Figure 5.7. The full models' architectures are listed in the evaluation section and they follow this validation of loss curves and parameters optimisation.

The model is trained for the optimal number of iterations with the generated batch of pairs as described in Algorithm 5.2. The number of iterations (2000, in this case) is decided by monitoring the loss curves after performing parameter optimisation. The $batch\_size = 30,000$ is based on the literature recommendation for the advisable Siamese network training batch size [277, 268]. It is important to note that the classes are equally represented in both the training and testing batches and that pair uniqueness is guaranteed. A "set" data structure is used so that a pair is added to the batch of pairs unless that pair is already contained within the set, as demonstrated in Algorithm 5.2. As outlined in the algorithm, the dataset should have at least 3 classes, otherwise, the Siamese network model converges to a 50% similarity output and fails to train adequately. This is because when there are only two classes, A and B for example, the instances have a 0.5 probability of being similar [(A,A) or (B,B)] or dissimilar [(A,B), (B,A)]. Since the dissimilar pairs resemble the same combination, the similarity learning will converge to a 50% output (0.5 probability).

Algorithm 5.2 Siamese Network: Generate Training Batch

Input: Dataset of N classes, Batch Size
Output: Batch of similar and dissimilar pairs
and associated labels (0: dissimilar, 1: similar)

1: **function** GETTRAININGBATCH(batch_size)
2: $num\_similar\_pairs = batch\_size/2$
3: $num\_dissimilar\_pairs = batch\_size/2$
4: $num\_similar\_pairs\_per\_class = num\_similar\_pairs/N$
5: $all\_combinations = combinations(N)$
6: $num\_dissimilar\_pairs\_per\_combination$
$= num\_dissimilar\_pairs/len(all\_combinations)$
7: $pairs\_set \leftarrow \{\}$
8: **for** c in N **do**
9: **for** $i = 0$ to $num\_similar\_pairs\_per\_class$ **do**
10: $(ins_1, ins_2) \leftarrow$ 2 random instances $\in c\_training$
11: **if** $(ins_1, ins_2) \in pairs\_set$ **then**
12: **go to** 10
13: **end if**
14: $pairs[i] \leftarrow \{ins_1, ins_2\}$
15: $pairs\_set.add(\{ins_1, ins_2\})$
16: **end for**
17: **end for**
18: **for** c_1, c_2 in $all\_combinations$ **do**
19: **for** $i = 0$ to $num\_dissimilar\_pairs\_per\_combination$ **do**
20: $ins_1 \leftarrow$ random instance $\in c_1\_training$
21: $ins_2 \leftarrow$ random instance $\in c_2\_training$
22: **if** $(ins_1, ins_2) \in pairs\_set$ **then**
23: **go to** 20
24: **end if**
25: $pairs[i] \leftarrow \{ins_1, ins_2\}$
26: $pairs\_set.add(\{ins_1, ins_2\})$
27: **end for**
28: **end for**
29: $targets[0..batch\_size/2] \leftarrow 1$ ▷ Similar
30: $targets[batch\_size/2..batch\_size] \leftarrow 0$ ▷ Dissimilar
31: **return** $pairs, targets$
32: **end function**

For the evaluation (Algorithm 5.3), an equal number of instances are used from each class. For each new instance, a pair is selected for each class using the new instance and a random instance from that class. The similarity is calculated for each pair and the pair with the closest similarity contributes to the classification result. This process is performed j times and voting is used to collate the results ($j \in 1, 5, 10, 15, 20, 25, 30$).

Algorithm 5.3 Siamese Network: Evaluate Classification

Input: Trained Siamese Network, Batch Size
Output: Classification Accuracy

function EVALCLASSIFICATION(batch_size)
    $n\_correct \leftarrow 0$
    $num\_per\_class \leftarrow batch\_size/N$
    for c in N **do**
        for $i = 0$ to num_per_class **do**
            $ins_1 \leftarrow$ random instance $\in c\_testing$
            for $j = 0$ to 5 **do**
                $pairs \leftarrow (ins_1,$ random instance $x \forall x \in N)$
                $similarities \leftarrow model.predict(pairs)$
                $votes[argmin(similarities)] += 1$
            end for
            if $argmax(votes) == c$ **then**
                $n\_correct += n\_correct + 1$
                $confusion\_matrix[c, argmax(votes)] += 1$
            end if
        end for
    end for
    $accuracy = n\_correct * 100/batch\_size$
    return $accuracy, confusion\_matrix$
end function

5.5.2 Experiments and Results

The evaluation specifies how accurately the network can classify based on learning similarities using a few samples from each class. The optimal hyperparameters of the twin networks; ANN architecture (number of hidden layers and neurons), learning rate, etc. are as follows (**bold**: input, *italic*: output of Siamese network before similarity calculation, Dr: Dropout layer):

- Twin Networks Architecture:
  - SCADA: In(**10**) : *8*
  - CICIDS2017: In(**31**) : 25 : Dr(0.1) : 20 : Dr(0.05) : *15*
  - NSL-KDD - KDD Cup'99:

    In(**118**) : 98 : Dr(0.1) : 79 : Dr(0.1) : 59 : Dr(0.1) : 39 : Dr(0.1) : *20*
- Activation: Relu

- L2 regularisation: 0.001
- Optimiser: Adam
- Number of epochs: 2000
- Loss: Constructive loss [258]

5.5.2.1 SCADA Dataset Results

The SCADA dataset classification Confusion Matrix (CM) is presented in Table 5.5. A sample CM is presented in Table 5.4. Each row of the CM represents an actual class and each column represents a predicted class, or vice versa. For the normal class row, True Negative (TN) and False Positive (FP) are recorded, while for attack classes, True Positive (TP) and False Negative (FN) are recorded. An ideal CM would have a diagonal of 100%, where all classes' instances are correctly classified/labelled.

Table 5.4
Sample Confusion Matrix

	Predicted Class				
Correct	Normal	$Attack_1$	$Attack_2$	$Attack_3$	$Attack_4$
Normal	TN	FP_1	FP_2	FP_3	FP_4
$Attack_1$	FN_1	TP_{11}	TP_{12}	TP_{13}	TP_{14}
$Attack_2$	FN_2	TP_{21}	TP_{22}	TP_{23}	TP_{24}
$Attack_3$	FN_3	TP_{31}	TP_{32}	TP_{33}	TP_{34}
$Attack_4$	FN_4	TP_{41}	TP_{42}	TP_{43}	TP_{44}

As shown in Table 5.5, the overall accuracy is 76.06% with $j = 5$. However, it is seen that the classes either have high classification accuracy (reaching 100%) or a low accuracy (less than 50%). The model accuracy using different j pairs for voting is outlined in Table 5.6. It is important to highlight that a ZeroR [290] (baseline majority classifier) will result in a classification accuracy of 7.14%, by classifying all instances as the majority class. It should be noted as well that the classes in the raw input space, are highly overlapping and that the dataset only contains the registers readings from the PLC.

Table 5.5
Siamese Network: SCADA Classification Confusion Matrix ($j = 5$)

	Predicted Class														
Correct	Normal (S1)	Blocked measure 1 (S2)	Blocked measure 2 (S3)	DoS (S4)	Humidity (S5)	2 Floating objects (S6)	7 Floating objects (S7)	Person hitting high intensity (S8)	Person hitting med intensity (S9)	Person hitting low intensity (S10)	Plastic bag (S11)	Sensor failure (S12)	Spoofing (S13)	Wrong connection (S14)	Overall
S1	969 **(48.45%)**	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	4 *(0.2%)*	110 *(5.5%)*	294 *(14.7%)*	0 *(0%)*	0 *(0%)*	376 *(18.8%)*	172 *(8.6%)*	0 *(0%)*	75 *(3.75%)*	
S2	0 *(0%)*	2000 **(100%)**	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	
S3	0 *(0%)*	0 *(0%)*	2000 **(100%)**	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	
S4	0 *(0%)*	0 *(0%)*	0 *(0%)*	2000 **(100%)**	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	76.06%
S5	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	2000 **(100%)**	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	
S6	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	2 *(0.1%)*	1416 **(70.8%)**	6 *(0.3%)*	152 *(7.6%)*	0 *(0%)*	0 *(0%)*	111 *(5.55%)*	1 *(0.05%)*	312 *(15.6%)*	0 *(0%)*	
S7	91 *(4.55%)*	469 *(23.45%)*	477 *(23.85%)*	0 *(0%)*	0 *(0%)*	5 *(0.25%)*	628 **(31.4%)**	0 *(0%)*	0 *(0%)*	0 *(0%)*	287 *(14.35%)*	24 *(1.2%)*	1 *(0.05%)*	18 *(0.9%)*	
S8	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	2000 **(100%)**	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	
S9	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	2000 **(100%)**	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	
S10	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	2000 **(100%)**	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	
S11	312 *(15.6%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	3 *(0.15%)*	96 *(4.8%)*	122 *(6.1%)*	43 *(2.15%)*	0 *(0%)*	0 *(0%)*	1279 **(63.95%)**	122 *(6.1%)*	20 *(1%)*	3 *(0.15%)*	
S12	306 *(15.3%)*	0 *(0%)*	0 *(0%)*	31 *(1.55%)*	0 *(0%)*	149 *(7.45%)*	60 *(3%)*	486 *(24.3%)*	0 *(0%)*	0 *(0%)*	423 *(21.15%)*	337 **(16.85%)**	201 *(10.05%)*	7 *(0.35%)*	
S13	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	83 *(4.15%)*	447 *(22.35%)*	0 *(0%)*	9 *(0.45%)*	0 *(0%)*	0 *(0%)*	26 *(1.3%)*	29 *(1.45%)*	1406 **(70.3%)**	0 *(0%)*	
S14	115 *(5.75%)*	197 *(9.85%)*	0 *(0%)*	0 *(0%)*	1 *(0.05%)*	0 *(0%)*	29 *(1.45%)*	0 *(0%)*	328 *(16.4%)*	0 *(0%)*	55 *(2.75%)*	12 *(0.6%)*	0 *(0%)*	1263 **(63.15%)**	

Table 5.6
Siamese Network: SCADA Classification Accuracy Using Different j Votes

No Votes	Overall	Normal	
(j)	Accuracy	TNR	FPR
1	72.23%	34.3%	65.7%
5	76.06%	48.45%	51.55%
10	77.77%	45.35%	54.65%
15	78.6%	46.65%	53.35%
20	79.18%	47.3%	52.7%
25	79.06%	46.75%	53.25%
30	79.21%	45.7%	54.3%

This overlap was evident in Section 4.4 through the variant classification accuracies, which led to the use of classification confidence to reach a higher scenario classification accuracy. This classes overlap is demonstrated further here by fitting a k-NN model (with k=30, to compare with Siamese network when 30 pairs are used for majority voting) to the data at the input space. It is noted that the k-NN model calculates the

distance between instances in their input space, while the Siamese network similarity calculates the distance between the outputs of the twin networks. Figure 5.8 shows for each class c, the number of wrongly labelled classes for instances of c (for example, instances of "S6: 2 Floating Objects" are misclassified as 7 other classes for the k-NN and 3 for the Siamese network). From Figure 5.8 it can be observed that the Siamese network model has learned a transformation that reduces the overlap between classes, justifying the performance improvements for classes S5: Humidity and S8-S10: Person Hitting.

Furthermore, it can be seen that the classes which do not overlap (for example, S2, S3: Blocked Measures and S4: DoS), have high classification accuracy. Finally, for classes overlapping with more than 7 other classes, the Siamese network is able to reduce the number of misclassified classes resulting from the overlap. However, "pair similarity", solely, did not achieve high classification accuracy for these classes.

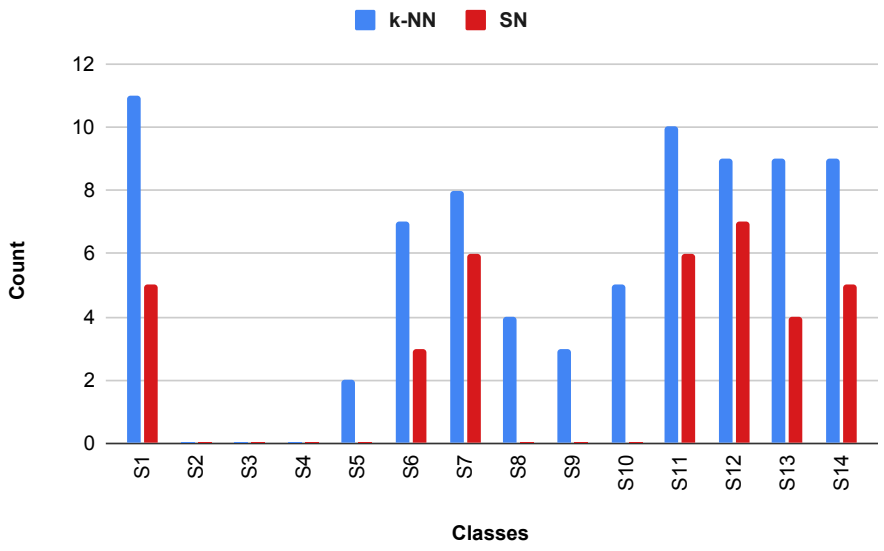


Figure 5.8
SCADA Dataset k-NN (k=30) and Siamese Network (SN) (30 pairs): Number of Wrong Associated Classes During Classification

5.5.2.2 CICIDS2017 Dataset Results

The CM of the classification for the CICIDS2017 is presented in Table 5.7. As presented, based only on pairs similarity, the overall accuracy is 83.74% with $j = 5$. The different attack classes accuracies are 96.08%, 75.17%, 80.05%, and 76.55%, respectively. Moreover, the low false negatives are presented in the first column. Also, a small FPR for Normal (0.05%, 2.6%, 1.87%, and 4.62%) for the attack classes respectively.

Table 5.7
Siamese Network: CICIDS2017 Classification Confusion Matrix ($j = 5$)

	Predicted Class					
Correct	Normal	DoS (Hulk)	DoS (Slowloris)	FTP	SSH	Overall
Normal	5452 **(90.87%)**	3 *(0.05%)*	156 *(2.6%)*	112 *(1.87%)*	277 *(4.62%)*	83.74%
DoS (Hulk)	139 *(2.32%)*	5765 **(96.08%)**	24 *(0.4%)*	13 *(0.22%)*	59 *(0.98%)*	
DoS (Slowloris)	914 *(15.23%)*	1 *(0.02%)*	4510 **(75.17%)**	71 *(1.18%)*	504 *(8.4%)*	
FTP	790 *(13.17%)*	2 *(0.03%)*	95 *(1.58%)*	4803 **(80.05%)**	310 *(5.17%)*	
SSH	973 *(16.22%)*	0 *(0%)*	227 *(3.78%)*	207 *(3.45%)*	4593 **(76.55%)**	

Table 5.8
Siamese Network: CICIDS2017 Classification Accuracy Using Different j Votes

No Votes	Overall	Normal	
(j)	Accuracy	TNR	FPR
1	74.55%	70.43%	29.57%
5	83.74%	90.87%	9.13%
10	84.54%	92.58%	7.42%
15	84.63%	93.07%	6.93%
20	84.69%	93.55%	6.45%
25	84.69%	93.73%	6.27%
30	84.71%	93.85%	6.15%

Table 5.8 lists the overall accuracy, TNR, and FPR when using different j pairs for voting. It is observed that using 5 pairs results in a distinctive rise in both the overall accuracy (from 74.55% to 83.74%) and the TNR (from 70.43% to 90.87%) than using 1 pair. The reason 1 pair performance is poor owes to the instance selection randomness. The probability of selecting a representable pair increases as j increases, therefore, increasing the likelihood of correct classification based on similarity. This random selection process is also affected by the instances variance and outliers.

5.5.2.3 KDD Cup'99 and NSL-KDD Datasets Results

The CM of the classification for the KDD Cup'99 dataset is presented in Table 5.9. As shown, the overall accuracy when $j = 5$ is 87.99% with a small portion of attack classes misclassified as normal (0.1%, 0.97%, 0.27%, and 8% for the attack classes respectively). Similar to the CICIDS2017 dataset, using 5 pairs results in a rise in the accuracy and TNR as outlined in Table 5.10.

Table 5.9
Siamese Network: KDD Cup'99 Classification Confusion Matrix ($j = 5$)

	Predicted Class					
Correct	Normal	DoS	Probe	R2L	U2R	Overall
Normal	4423 **(73.72%)**	9 *(0.15%)*	492 *(8.2%)*	979 *(16.32%)*	97 *(1.62%)*	87.99%
DoS	6 *(0.1%)*	5920 **(98.67%)**	64 *(1.07%)*	10 *(0.17%)*	0 *(0%)*	
Probe	58 *(0.97%)*	254 *(4.23%)*	5453 **(90.88%)**	222 *(3.7%)*	13 *(0.22%)*	
R2L	16 *(0.27%)*	0 *(0%)*	39 *(0.65%)*	5786 **(96.43%)**	159 *(2.65%)*	
U2R	480 *(8%)*	0 *(0%)*	685 *(11.42%)*	21 *(0.35%)*	4814 **(80.23%)**	

Table 5.10
Siamese Network: KDD Cup'99 Classification Accuracy Using Different j Votes

No Votes	Overall	Normal	
(j)	Accuracy	TNR	FPR
1	82.03%	69.27%	30.73%
5	87.99%	73.72%	26.28%
10	88.26%	73.67%	26.33%
15	88.29%	73.63%	26.37%
20	88.26%	73.65%	26.35%
25	88.23%	73.6%	26.4%
30	88.24%	73.6%	26.4%

Training the Siamese network model on the NSL-KDD dataset, which is an improved dataset based on the KDD Cup'99 (filtered where duplicates are removed), did show a minor rise in the classification results. The CM of the NSL-KDD dataset is presented in Table 5.11 and the different j votes performance is in Table 5.12. The overall accuracy increased to 91.01% compared to 87.99% for the KDD Cup'99 dataset.

Table 5.11
Siamese Network: NSL-KDD Classification Confusion Matrix ($j = 5$)

	Predicted Class					
Correct	Normal	DoS	Probe	R2L	U2R	Overall
Normal	5187 **(86.45%)**	47 *(0.78%)*	300 *(5%)*	315 *(5.25%)*	151 *(2.52%)*	91.01%
DoS	144 *(2.4%)*	5621 **(93.68%)**	217 *(3.62%)*	16 *(0.27%)*	2 *(0.03%)*	
Probe	159 *(2.65%)*	643 *(10.72%)*	5133 **(85.55%)**	44 *(0.73%)*	21 *(0.35%)*	
R2L	227 *(3.78%)*	0 *(0%)*	31 *(0.52%)*	5669 **(94.48%)**	73 *(1.22%)*	
U2R	214 *(3.57%)*	0 *(0%)*	92 *(1.53%)*	2 *(0.03%)*	5692 **(94.87%)**	

Table 5.12
Siamese Network: NSL-KDD Classification Accuracy Using Different j Votes

No Votes	Overall	Normal	
(j)	Accuracy	TNR	FPR
1	86.61%	80.47%	19.53%
5	91.01%	86.45%	13.55%
10	91.1%	86.45%	13.55%
15	91.17%	86.4%	13.6%
20	91.24%	86.47%	13.53%
25	91.26%	86.42%	13.58%
30	91.3%	86.53%	13.47%

By comparing the KDD Cup'99 dataset results to the NSL-KDD ones, the minor improvement can explained by the learning approach of the Siamese network. Since the Siamese network learns from similarities, rather than specific class features, it can overcome the balancing or duplicate issues. The randomisation of choosing the training batch pairs and ensuring the balanced representation of class pairs resolve this as well.

To the best of the author's knowledge, there are no manuscripts that use Siamese networks or leverage similarity-based One-Shot learning for IDS. However, the performance of recent articles that use the aforementioned datasets is outlined below. Comparing the performance of the models presented here with recent IDS models is not straightforward, yet, their performance aid in the interpretation of the different classes performance results.

Recent IDS articles evaluation is outlined in Table 5.13. These studies focus on multi-class attack classification and report explicit class metrics, not only the overall accuracy. It is important to note that by observing Table 5.13, it is evident that the overall classification accuracy is higher than each class performance. This is due to class imbalance problem. For example in [176], the TPR for the SSH and FTP attack classes in the CICIDS2017 dataset are 0% and 3.1%, respectively, while the overall accuracy is 96%. Similarly, the TPR for the R2L and U2R in the KDD Cup'99 dataset

is 24.3% and 15.5%, respectively, with an overall accuracy of 92.6%. Class imbalance problem is a common problem with datasets and is considered relative to the degree of imbalance, the overall dataset size, and the complexity of the data [291, 292]. Common approaches to overcome class imbalance are upsizing, downsizing, and altering the contribution of misclassifying under-sampled and over-sampled classes to the overall accuracy [291, 292]. None of these methods have been used in the papers discussed in Table 5.13, which resulted in both a gap between classes detection accuracy and overall accuracy, and the misleading overall accuracy results. It is important to note that the class imbalance problem did not pose a problem for the evaluation presented in this Chapter. This is due to the fact that equal number of pairs are randomly selected from a pool of instances, which ensures balance in training and testing.

With regards to the results presented in this section and those in Table 5.13, KDD Cup'99 overall accuracy using the Siamese network model reaches 88% compared to 92.6% in [176] and 99.8% in [182]. However, by analysing the TPR of the different classes, it is observed that the Siamese network experience higher TPR for the attack classes. For example, the Siamese network model TPR of DoS, Probe, R2L and U2R are 98.67%, 90.88%, 96.43%, and 80.23% compared to 99.9%, 98.9%, 96.9%, and 75% in [182], and 93.9%, 73.2%, 24.3% , and 15.5% in [176], respectively. Similarly, the overall accuracy of the NSL-KDD reaches 91% for the Siamese network model compared with 77.8% in [176] and 83.83% [293]. The TPR of DoS, Probe, R2L, and U2R when using Siamese Network are 93.68%, 85.55%, 94.48%, and 94.87% compared with 97.42%, 96.51%, 68.53%, and 95.14% in [177] and 86.63%, 83.73%, 35.15%, and 23.5% in [294]. Finally, the CICIDS2017 overall accuracy reaches 84% using the Siamese network model, compared with 96% in [184]. The TPR of FTP and SSH classes using the Siamese network model is 80.05% and 76.55% compared with 98% and 77% in [184] and 0% and 3.1% in [176].

Table 5.13
Recent IDS Studies for Multi-Class Classification Performance

Year/ Reference	ML Technique	Metric	Result
CICIDS2017			
2019 / [176]	Deep Neural Network with 1 Layer	Accuracy	Overall: 96%
		TPR	Normal: 64.6% SSH: 0% FTP:3.1% DDoS: 9.5%
2020 / [184]	Multi-layer Perceptron	Recall	SSH: 98% FTP: 77%
KDD Cup'99			
2019 / [176]	Deep Neural Network with 1 Layer	Accuracy	Overall: 92.6%
		TPR	Normal: 99.4% DoS: 93.9% Probe: 73.2% R2L: 24.3% U2R: 15.5%
2020 / [182]	Ensemble Model	Accuracy	Overall: 99.8%
		TPR	Normal: 99.99% DoS: 99.99% Probe: 98.9% R2L: 96.9% U2R: 75%
NSL-KDD			
2019 / [177]	Multi-layer Perceptron	Accuracy	Normal: 87.31% DoS: 97.42% Probe: 96.51% R2L: 68.53% U2R: 95.14%
2019 / [293]	Ensemble model	Overall	$KDDTest^{+}$: 83.83% $KDDTest^{-21}$: 78.33%
2019 / [176]	Deep Neural Network with 1 Layer	Accuracy	Overall: 77.8%
		TPR	Normal: 97.3% DoS: 77.7% Probe: 61% R2L: 43.3% U2R: 24.1%
2020 / [294]	Multi-CNN	Recall	$\underline{KDDTest^{+}}$: Normal: 91.19% DoS: 86.63% Probe: 83.73% R2L: 35.15% U2R: 23.50% $\underline{KDDTest^{-21}}$: Normal: 62.08% DoS: 77.04% Probe: 82.60% R2L: 35.15% U2R: 23.50%

5.6 Scenario 2: One-Shot Detection

In this section, the Siamese network is used as a One-Shot learning architecture. The experiment evaluates the Siamese network performance on classifying a new cyber attack class without the need for retraining. This new class is represented with a few labelled samples. The experiment evaluates how accurate the similarity measure is, showing the capability of the Siamese network to find similarity between pairs of classes that were not used during the training process.

5.6.1 Methodology

Figure 5.9 shows the process of building the intrusion detection model and then evaluating it with an additional class that is not used during training. The process is similar to the one presented in Figure 5.4. However, the difference between both is that a class e is excluded from the training classes as shown in Figure 5.9-1. Class e is used to mimic a real-life situation in which a new attack is detected and only a few samples of it are available. In situations where a few instances are not enough to retrain a traditional IDS, there is still need to classify this new attack until enough samples become available for retraining.

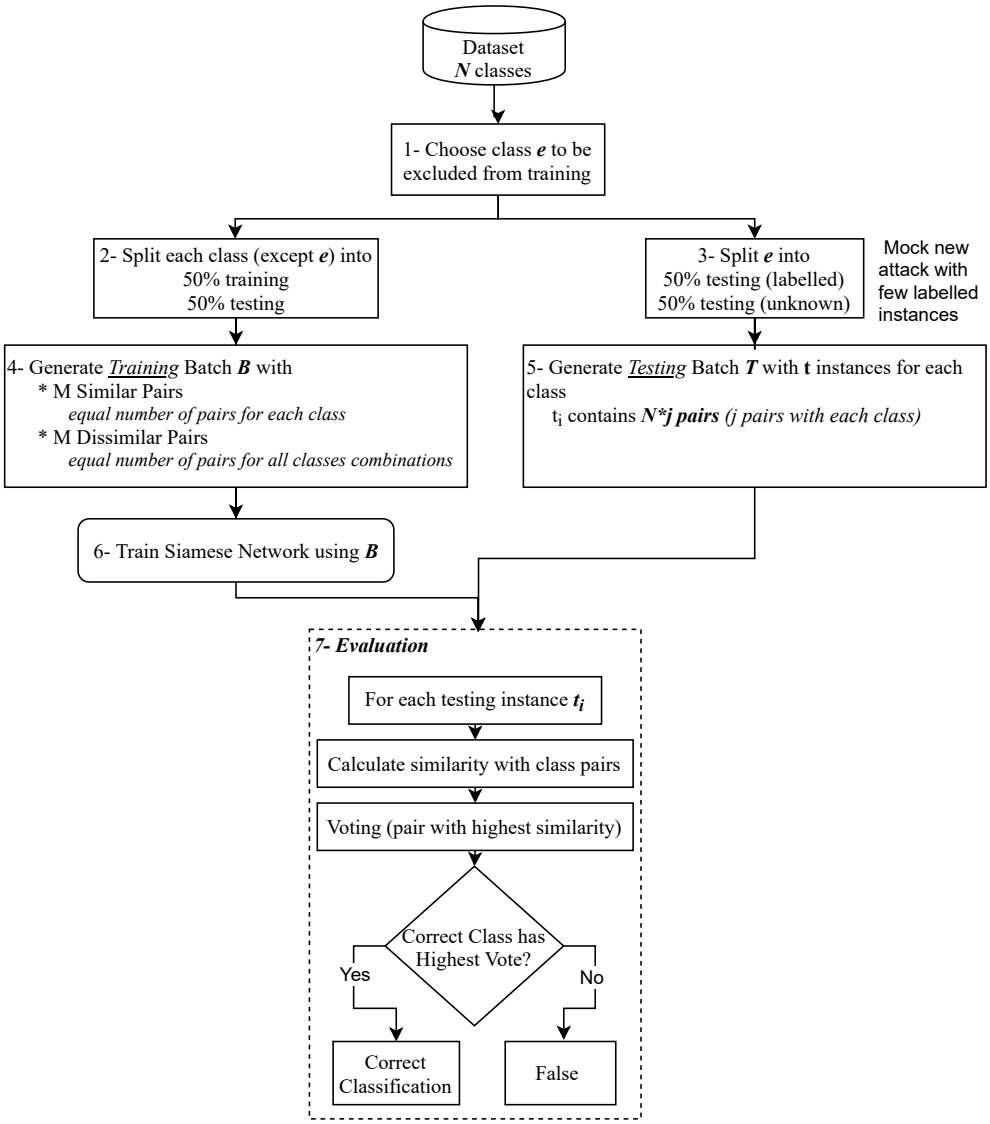


Figure 5.9
Siamese Network for Intrusion Detection (One-Shot Learning)

Moreover, Algorithm 5.4 summarises the overall process of training and testing the Siamese network model. The difference is in the evaluation process which is outlined in Algorithm 5.5.

The instances of class e are split such that 50% represents the labelled samples (i.e., mock adding them to the pool of instances for Siamese network to pair with during testing) and the other 50% are used as new instances to evaluate the accuracy.

Algorithm 5.4 Siamese Network: Usage Scenario 2 Train and Test Algorithm

Input: Attacks Dataset
Output: Trained Siamese Network Evaluation

Ensure: $dataset = \{c_1, c_2, \ldots, c_n : n \geq 3\}$
1: $train\_batch\_size, test\_batch\_size \leftarrow 30,000$
2: $n\_epochs \leftarrow 2000$
3: $excluded\_class =$ random class e s.th. $e \in dataset$
4: $training\_classes = dataset - e$
5: $training = 50\%\ c_i\ \forall c_i \in training\_classes$
6: $testing = dataset \cap \overline{training}$
7: $batch \leftarrow$ GETTRAININGBATCH($train\_batch\_size$)
8: Build Siamese Network with Random Weights
9: **for** $i = 0$ to $n\_iterations$ **do**
10: Update Siamese Network Weights based on $batch$
11: **end for**
12: EVALUATEONESHOT(test_batch_size)

Algorithm 5.5 Siamese Network: Evaluate One-Shot Model

Input: Trained Siamese Network, Batch Size, Excluded Class (e)
Output: Accuracy

1: **function** EVALUATEONESHOT(batch_size)
2: $n\_correct \leftarrow 0$
3: $num\_per\_class \leftarrow batch\_size/N$
4: $K \leftarrow N - e$
5: **for** c in N **do**
6: **for** $i = 0$ to num_per_class **do**
7: **if** $c == e$ **then**
8: $ins_1 \leftarrow$ random instance $\in e\_unlabelled$
9: **else**
10: $ins_1 \leftarrow$ random instance $\in c\_testing$
11: **end if**
12: **for** $j = 0$ to 5 **do**
13: $pairs \leftarrow (ins_1$, random instance $x \forall x \in K)$
14: $pairs.append(ins_1$, random instance $\in e\_labelled$
15: $similarities \leftarrow model.predict(pairs)$
16: $votes[argmin(similarities)] += 1$
17: **end for**
18: **if** $argmax(votes) == c$ **then**
19: $n\_correct += n\_correct + 1$
20: **end if**
21: $confusion\_matrix[c, argmax(votes)] += 1$
22: **end for**
23: **end for**
24: $accuracy = n\_correct * 100/batch\_size$
25: **return** $accuracy, confusion\_matrix$
26: **end function**

5.6.2 Experiments and Results

The evaluation specifies how accurately the Siamese network can utilise similarity learning to classify: (i) the classes that are used in training and (ii) a new class that is not used during training using few instances.

For the One-Shot evaluation, multiple experiments, specifically $N - 1$ where N is the number of classes, are conducted to evaluate the performance of the Siamese network when using a different set of attack classes for training and evaluation. In each experiment, a different class of the dataset is excluded, one at a time.

5.6.2.1 SCADA Dataset Results

The SCADA One-Shot experimental results follow the same behaviour as discussed in Section 5.5.2.1 where classes overlapping with less than seven other classes have a high classification accuracy. The classes that are not overlapping with others, (for example, DoS and Blocked Measures) show high TPR when introduced after training (acting as the new class). Table 5.14 and Table 5.15 list the CM and the different pairs performance for the first Blocked Measure class, while Table 5.16 and Table 5.17 list for the DoS attack class. The detection rate of Blocked Measure 1 when 1 pair is used is 100%, as shown in Table 5.15. Similarly, the detection rate of DoS is 100% as shown in Table 5.17. Both the CM of One-Shot learning when excluding different classes, and the CM of classification, demonstrate the disparity of detection rates between classes, showing the high rates for classes that do not overlap or overlap with less than seven classes, and low rates otherwise.

Table 5.14
Siamese Network: SCADA One-Shot Confusion Matrix ($j = 5$) (Blocked Measure 1 excluded from Training)

	Predicted Class														
Correct	Normal (S1)	Blocked measure 1 (S2)	Blocked measure 2 (S3)	DoS (S4)	Humidity (S5)	2 Floating objects (S6)	7 Floating objects (S7)	Person hitting high intensity (S8)	Person hitting med intensity (S9)	Person hitting low intensity (S10)	Plastic bag (S11)	Sensor failure (S12)	Spoofing (S13)	Wrong connection (S14)	Overall
S1	1181 **(59.05%)**	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	393 *(19.65%)*	0 *(0%)*	0 *(0%)*	225 *(11.25%)*	201 *(10.05%)*	0 *(0%)*	0 *(0%)*	
S2	0 *(0%)*	2000 **(100%)**	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	
S3	0 *(0%)*	0 *(0%)*	2000 **(100%)**	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	
S4	0 *(0%)*	0 *(0%)*	0 *(0%)*	2000 **(100%)**	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	78.86%
S5	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	2000 **(100%)**	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	
S6	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	1980 **(99%)**	20 *(1%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	
S7	0 *(0%)*	0 *(0%)*	565 *(28.25%)*	0 *(0%)*	0 *(0%)*	82 *(4.1%)*	590 **(29.5%)**	0 *(0%)*	0 *(0%)*	0 *(0%)*	11 *(0.55%)*	6 *(0.3%)*	742 *(37.1%)*	4 *(0.2%)*	
S8	8 *(0.4%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	1987 **(99.35%)**	0 *(0%)*	0 *(0%)*	5 *(0.25%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	
S9	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	2000 **(100%)**	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	
S10	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	2000 **(100%)**	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	
S11	489 *(24.45%)*	37 *(1.85%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	19 *(0.95%)*	4 *(0.2%)*	357 *(17.85%)*	0 *(0%)*	0 *(0%)*	764 **(38.2%)**	92 *(4.6%)*	238 *(11.9%)*	0 *(0%)*	
S12	476 *(23.8%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	120 *(6%)*	0 *(0%)*	7 *(0.35%)*	384 *(19.2%)*	0 *(0%)*	0 *(0%)*	217 *(10.85%)*	290 **(14.5%)**	222 *(11.1%)*	284 *(14.2%)*	
S13	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	25 *(1.25%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	6 *(0.3%)*	1 *(0.05%)*	1966 **(98.3%)**	2 *(0.1%)*	
S14	0 *(0%)*	323 *(16.15%)*	0 *(0%)*	0 *(0%)*	51 *(2.55%)*	0 *(0%)*	11 *(0.55%)*	31 *(1.55%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	71 *(3.55%)*	189 *(9.45%)*	1324 **(66.2%)**	

Table 5.15
Siamese Network: SCADA One-Shot Accuracy (Blocked Measure 1 excluded from Training) Using Different j Votes

No Votes	Overall	New Class (S2)		Normal	
(j)	Accuracy	TPR	FNR	TNR	FPR
1	76.65%	100%	0%	40.65%	59.35%
5	78.86%	100%	0%	59.05%	40.95%
10	79.62%	100%	0%	63.5%	36.5%
15	80.13%	100%	0%	66.5%	33.5%
20	80.2%	100%	0%	67.85%	32.15%
25	80.21%	100%	0%	67.7%	32.3%
30	80.28%	100%	0%	69.05%	30.95%

Table 5.16
Siamese Network: SCADA One-Shot Confusion Matrix ($j = 5$) (DoS excluded from Training)

	Predicted Class														
Correct	Normal (S1)	Blocked measure 1 (S2)	Blocked measure 2 (S3)	DoS (S4)	Humidity (S5)	2 Floating objects (S6)	7 Floating objects (S7)	Person hitting high intensity (S8)	Person hitting med intensity (S9)	Person hitting low intensity (S10)	Plastic bag (S11)	Sensor failure (S12)	Spoofing (S13)	Wrong connection (S14)	Overall
S1	1297 **(64.85%)**	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	19 *(0.95%)*	78 *(3.9%)*	95 *(4.75%)*	0 *(0%)*	0 *(0%)*	326 *(16.3%)*	162 *(8.1%)*	12 *(0.6%)*	11 *(0.55%)*	
S2	0 *(0%)*	2000 **(100%)**	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	
S3	0 *(0%)*	0 *(0%)*	2000 **(100%)**	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	
S4	0 *(0%)*	0 *(0%)*	0 *(0%)*	2000 **(100%)**	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	80.09%
S5	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	2000 **(100%)**	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	
S6	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	1971 **(98.55%)**	17 *(0.85%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	12 *(0.6%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	
S7	273 *(13.65%)*	0 *(0%)*	454 *(22.7%)*	0 *(0%)*	0 *(0%)*	87 *(4.35%)*	697 **(34.85%)**	0 *(0%)*	0 *(0%)*	0 *(0%)*	332 *(16.6%)*	16 *(0.8%)*	135 *(6.75%)*	6 *(0.3%)*	
S8	1 *(0.05%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	1999 **(99.95%)**	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	
S9	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	2000 **(100%)**	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	
S10	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	2000 **(100%)**	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	
S11	535 *(26.75%)*	49 *(2.45%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	158 *(7.9%)*	164 *(8.2%)*	39 *(1.95%)*	0 *(0%)*	0 *(0%)*	726 **(36.3%)**	87 *(4.35%)*	229 *(11.45%)*	13 *(0.65%)*	
S12	674 *(33.7%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	410 *(20.5%)*	28 *(1.4%)*	25 *(1.25%)*	64 *(3.2%)*	0 *(0%)*	0 *(0%)*	253 *(12.65%)*	317 **(15.85%)**	227 *(11.35%)*	2 *(0.1%)*	
S13	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	2 *(0.1%)*	2 *(0.1%)*	1996 **(99.8%)**	0 *(0%)*	
S14	32 *(1.6%)*	319 *(15.95%)*	0 *(0%)*	0 *(0%)*	26 *(1.3%)*	15 *(0.75%)*	24 *(1.2%)*	27 *(1.35%)*	0 *(0%)*	0 *(0%)*	130 *(6.5%)*	5 *(0.25%)*	0 *(0%)*	1422 **(71.1%)**	

Table 5.17
Siamese Network: SCADA One-Shot Accuracy (DoS excluded from Training) Using Different j Votes

No Votes	Overall	New Class (S4)		Normal	
(j)	Accuracy	TPR	FNR	TNR	FPR
1	77.58%	100%	0%	40.15%	59.85%
5	80.09%	100%	0%	64.85%	35.15%
10	81.25%	100%	0%	70.9%	29.1%
15	82.13%	100%	0%	75.9%	24.1%
20	82.34%	100%	0%	76.85%	23.15%
25	82.78%	100%	0%	80.15%	19.85%
30	82.88%	100%	0%	81.4%	18.6%

Classes where the overlapping covers less than half of the other classes (humidity and different hitting intensities, for example) show high accuracy TPR when introduced after training. This is shown in Table 5.18 and Table 5.19 for person hitting with high intensity class. The detection rate rises from 55.05% when using one pair to 71.95% when using 5 pairs and reaches its highest of 87% when using 30 pairs.

Finally, classes with high overlap encounter low TPR, similar to their classification accuracy whether they are introduced during training or post training. For example, Wrong Connection is not detected when introduced after training, and all its instances are misclassified as other anomaly scenarios. For completeness, the full CM tables are listed in Appendix D.1.

Table 5.18
Siamese Network: SCADA One-Shot Confusion Matrix ($j = 5$) (Person Hitting High Intensity excluded from Training)

	Predicted Class														
Correct	Normal (S1)	Blocked measure 1 (S2)	Blocked measure 2 (S3)	DoS (S4)	Humidity (S5)	2 Floating objects (S6)	7 Floating objects (S7)	Person hitting high intensity (S8)	Person hitting med intensity (S9)	Person hitting low intensity (S10)	Plastic bag (S11)	Sensor failure (S12)	Spoofing (S13)	Wrong connection (S14)	Overall
S1	539 **(26.95%)**	0 *(0%)*	0 *(0%)*	0 *(0%)*	103 *(5.15%)*	416 *(20.8%)*	33 *(1.65%)*	623 *(31.15%)*	0 *(0%)*	0 *(0%)*	183 *(9.15%)*	66 *(3.3%)*	29 *(1.45%)*	8 *(0.4%)*	
S2	0 *(0%)*	2000 **(100%)**	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	
S3	0 *(0%)*	0 *(0%)*	2000 **(100%)**	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	
S4	0 *(0%)*	0 *(0%)*	0 *(0%)*	1998 **(99.9%)**	2 *(0.1%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	66.69%
S5	0 *(0%)*	0 *(0%)*	0 *(0%)*	393 *(19.65%)*	1607 **(80.35%)**	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	
S6	182 *(9.1%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	1495 **(74.75%)**	5 *(0.25%)*	19 *(0.95%)*	0 *(0%)*	0 *(0%)*	45 *(2.25%)*	55 *(2.75%)*	199 *(9.95%)*	0 *(0%)*	
S7	58 *(2.9%)*	28 *(1.4%)*	533 *(26.65%)*	153 *(7.65%)*	325 *(16.25%)*	23 *(1.15%)*	411 **(20.55%)**	60 *(3%)*	3 *(0.15%)*	148 *(7.4%)*	105 *(5.25%)*	30 *(1.5%)*	104 *(5.2%)*	19 *(0.95%)*	
S8	295 *(14.75%)*	2 *(0.1%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	116 *(5.8%)*	1 *(0.05%)*	1439 **(71.95%)**	0 *(0%)*	33 *(1.65%)*	28 *(1.4%)*	71 *(3.55%)*	6 *(0.3%)*	9 *(0.45%)*	
S9	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	2000 **(100%)**	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	
S10	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	2000 **(100%)**	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	
S11	293 *(14.65%)*	0 *(0%)*	0 *(0%)*	10 *(0.5%)*	5 *(0.25%)*	132 *(6.6%)*	32 *(1.6%)*	212 *(10.6%)*	0 *(0%)*	0 *(0%)*	574 **(28.7%)**	54 *(2.7%)*	630 *(31.5%)*	58 *(2.9%)*	
S12	318 *(15.9%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	80 *(4%)*	200 *(10%)*	20 *(1%)*	642 *(32.1%)*	0 *(0%)*	0 *(0%)*	175 *(8.75%)*	206 **(10.3%)**	357 *(17.85%)*	2 *(0.1%)*	
S13	14 *(0.7%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	200 *(10%)*	29 *(1.45%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	544 *(27.2%)*	122 *(6.1%)*	1089 **(54.45%)**	2 *(0.1%)*	
S14	74 *(3.7%)*	363 *(18.15%)*	0 *(0%)*	100 *(5%)*	16 *(0.8%)*	29 *(1.45%)*	11 *(0.55%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	60 *(3%)*	13 *(0.65%)*	18 *(0.9%)*	1316 **(65.8%)**	

Table 5.19
Siamese Network: SCADA One-Shot Accuracy (Person Hitting High Intensity excluded from Training) Using Different j Votes

No Votes	Overall	New Class (S8)		Normal	
(j)	Accuracy	TPR	FNR	TNR	FPR
1	63.63%	55.05%	17.65%	21.65%	78.35%
5	66.69%	71.95%	14.75%	26.95%	73.05%
10	67.85%	79.9%	7.65%	24.05%	75.95%
15	68.22%	83.45%	5.15%	22.1%	77.9%
20	68.34%	85.6%	3.2%	21.45%	78.55%
25	68.43%	86.6%	2.95%	21%	79%
30	68.47%	87%	2.8%	20.5%	79.5%

5.6.2.2 CICIDS2017 Dataset Results

The CM of the CICIDS2017 dataset when excluding SSH class is presented in Table 5.20 and excluding FTP in Table 5.22. The overall accuracy is 81.28% and 82.5%, respectively. The overall accuracy demonstrates that the network performance was not disturbed by the attack class addition post training when compared to 83.74% classification accuracy when all classes are used in training and testing (Table 5.7). It is important to note that the new attack class performance is 73.03% and 70.03% for SSH and FTP, respectively when using 5 pairs. Moreover, the added class demonstrates low False Negative Rate (FNR), specifically 8.63% and 15.4% for FTP and SSH, respectively. .

Table 5.21 and Table 5.23 present the evaluation results, showing the performance impact the number of labelled samples (j) of the new attack class e has. This is shown in terms of overall accuracy, new attack TPR, FNR, Normal TNR and FPR, using j instances for majority voting, where $j \in 1, 5, 10, 15, 20, 25, 30$.

Using five labelled instances of the new attack class results in an increase in the overall accuracy and the TPR accompanied with a drop in the FNR. Using only 1 labelled instance demonstrates a comparably poorer performance owing to the instance selection randomness, which could result in either a good or a bad class representative. However, using five random labelled instances boosts performance, reinforcing the importance of having distinctive class representatives. This is further demonstrated in the steady rise of the TPR when more pairs are used. For example, the SSH TPR rises as follows; 64.10%, 73.03%, 77.82%, 78.33%, 78.30%, and 78.45% for $j \in (1, 5, 10, 15, 20, 25, 30)$, respectively. In a similar fashion, the TPR of the FTP class rises from 56.65% when using one pair to 78.48% when using 30 pairs.

Table 5.20
Siamese Network: CICIDS2017 One-Shot Confusion Matrix ($j = 5$) (SSH excluded from training)

Correct	Predicted Class: Normal	DoS (Hulk)	DoS (Slowloris)	FTP	SSH	Overall
Normal	4711 **(78.52%)**	9 *(0.15%)*	103 *(1.72%)*	148 *(2.47%)*	1029 *(17.15%)*	81.28%
DoS (Hulk)	93 *(1.55%)*	5745 **(95.75%)**	33 *(0.55%)*	43 *(0.72%)*	86 *(1.43%)*	
DoS (Slowloris)	507 *(8.45%)*	0 *(0%)*	4668 **(77.8%)**	143 *(2.38%)*	682 *(11.37%)*	
FTP	643 *(10.72%)*	1 *(0.02%)*	127 *(2.12%)*	4879 **(81.32%)**	350 *(5.83%)*	
SSH	924 *(15.4%)*	34 *(0.57%)*	310 *(5.17%)*	350 *(5.83%)*	4382 **(73.03%)**	

Table 5.21
Siamese Network: CICIDS2017 One-Shot Accuracy (SSH excluded from Training) Using Different j Votes

No Votes	Overall	New Class (SSH)		Normal	
(j)	Accuracy	TPR	FNR	TNR	FPR
1	72.72%	64.10%	16.43%	63.35%	36.65%
5	81.28%	73.03%	15.40%	78.52%	21.48%
10	82.56%	77.82%	13.40%	79.95%	20.05%
15	82.58%	78.43%	13.03%	79.92%	20.08%
20	82.49%	78.33%	13.18%	79.97%	20.03%
25	82.43%	78.30%	13.25%	79.78%	20.22%
30	82.49%	78.45%	13.13%	79.97%	20.03%

Table 5.22
Siamese Network: CICIDS2017 One-Shot Confusion Matrix ($j = 5$) (FTP excluded from training)

Correct	Predicted Class: Normal	DoS (Hulk)	DoS (Slowloris)	FTP	SSH	Overall
Normal	5231 **(87.18%)**	3 *(0.05%)*	152 *(2.53%)*	189 *(3.15%)*	425 *(7.08%)*	82.5%
DoS (Hulk)	70 *(1.17%)*	5755 **(95.92%)**	48 *(0.8%)*	15 *(0.25%)*	112 *(1.87%)*	
DoS (Slowloris)	424 *(7.07%)*	1 *(0.02%)*	4433 **(73.88%)**	485 *(8.08%)*	657 *(10.95%)*	
FTP	518 *(8.63%)*	1 *(0.02%)*	659 *(10.98%)*	4202 **(70.03%)**	620 *(10.33%)*	
SSH	546 *(9.1%)*	3 *(0.05%)*	198 *(3.3%)*	124 *(2.07%)*	5129 **(85.48%)**	

Table 5.23
Siamese Network: CICIDS2017 One-Shot Accuracy (FTP excluded from Training) Using Different j Votes

No Votes	Overall	New Class (FTP)		Normal	
(j)	Accuracy	TPR	FNR	TNR	FPR
1	72.91%	59.65%	8.03%	72.83%	27.17%
5	82.5%	70.03%	8.63%	87.18%	12.82%
10	84.57%	72.80%	8.32%	87.70%	12.30%
15	85.47%	76.72%	8.12%	87.40%	12.60%
20	85.78%	77.58%	8.10%	87.23%	12.77%
25	85.86%	78.27%	8.10%	86.92%	13.08%
30	85.94%	78.48%	8.00%	86.73%	13.27%

For transparency and reproducibility, the rest of the CICIDS2017 performance evaluation tables are listed in Appendix D.2 and they follow similar performance behaviour. DoS (Slowloris) result tables is listed in Table D.21 and Table D.22. The TPR rises from 50.97% when using one pair to 72.82% when using 30 pairs. DoS (Hulk) results are listed in Table D.23 and Table D.24, where the TPR rises from 91.07% when using one pair to 95.18% when using 30 pairs.

5.6.2.3 KDD Cup'99 and NSL-KDD Datasets Results

The CM of the KDD Cup'99 and NSL-KDD datasets One-Shot when DoS class is excluded from training are presented in Table 5.24 and Table 5.26, respectively. As observed, the overall accuracies are 76.67% and 77.99%, respectively. It is important to note, however, that the FNR for the new class (i.e. DoS) are 26.38% for the KDD Cup'99 and 9.87% for the NSL-KDD and the TPR are 40.28% and 78.87% respectively. These percentages clearly demonstrate that the NSL-KDD results are higher because it is an enhanced version of the KDD Cup'99. Given that the new class is not used in training, having a better representation of instances shows a better performance (i.e., NSL-KDD outperforms KDD Cup'99).

Table 5.24
Siamese Network: KDD Cup'99 One-Shot Confusion Matrix ($j = 5$) (DoS excluded from Training)

	Predicted Class					
Correct	Normal	DoS	Probe	R2L	U2R	Overall
Normal	4562 **(76.03%)**	243 *(4.05%)*	522 *(8.7%)*	579 *(9.65%)*	94 *(1.57%)*	76.67%
DoS	1583 *(26.38%)*	2417 **(40.28%)**	1831 *(30.52%)*	168 *(2.8%)*	1 *(0.02%)*	
Probe	159 *(2.65%)*	214 *(3.57%)*	5367 **(89.45%)**	242 *(4.03%)*	18 *(0.3%)*	
R2L	56 *(0.93%)*	275 *(4.58%)*	10 *(0.17%)*	5571 **(92.85%)**	88 *(1.47%)*	
U2R	17 *(0.28%)*	205 *(3.42%)*	655 *(10.92%)*	40 *(0.67%)*	5083 **(84.72%)**	

Table 5.25
Siamese Network: KDD Cup'99 One-Shot Accuracy (DoS excluded from Training) Using Different j Votes

No Votes	Overall	New Class (DoS)		Normal	
(j)	Accuracy	TPR	FNR	TNR	FPR
1	66.89%	41.67%	22.50%	66.35%	33.65%
5	76.67%	40.28%	26.38%	76.03%	23.97%
10	77.57%	40.07%	27.25%	76.10%	23.90%
15	77.67%	39.90%	27.32%	76.02%	23.98%
20	77.68%	39.93%	27.38%	76.02%	23.98%
25	77.68%	39.87%	27.40%	76.07%	23.93%
30	77.68%	39.88%	27.40%	76.03%	23.97%

Table 5.26
Siamese Network: NSL-KDD Cup'99 One-Shot Confusion Matrix ($j = 5$) (DoS excluded from Training)

	Predicted Class					
Correct	Normal	DoS	Probe	R2L	U2R	Overall
Normal	5593 **(93.22%)**	61 *(1.02%)*	136 *(2.27%)*	122 *(2.03%)*	88 *(1.47%)*	77.99%
DoS	592 *(9.87%)*	4732 **(78.87%)**	653 *(10.88%)*	12 *(0.2%)*	11 *(0.18%)*	
Probe	67 *(1.12%)*	3305 *(55.08%)*	2595 **(43.25%)**	19 *(0.32%)*	14 *(0.23%)*	
R2L	212 *(3.53%)*	7 *(0.12%)*	27 *(0.45%)*	5692 **(94.87%)**	62 *(1.03%)*	
U2R	486 *(8.1%)*	6 *(0.1%)*	31 *(0.52%)*	693 *(11.55%)*	4784 **(79.73%)**	

Table 5.27
Siamese Network: NSL-KDD One-Shot Accuracy (DoS excluded from Training) Using Different j Votes

No Votes	Overall	New Class (DoS)		Normal	
(j)	Accuracy	TPR	FNR	TNR	FPR
1	72.75%	67.35%	9.05%	84.87%	15.13%
5	77.99%	78.87%	9.87%	93.22%	6.78%
10	77.7%	84.62%	9.87%	93.35%	6.65%
15	79.05%	83.78%	9.87%	93.32%	6.68%
20	78.63%	85.25%	9.87%	93.37%	6.63%
25	79.49%	84.62%	9.87%	93.35%	6.65%
30	79.12%	85.37%	9.87%	93.35%	6.65%

For transparency and reproducibility, the rest of the KDD Cup'99 and NSL-KDD dataset result tables are listed in Appendix D.3 and Appendix D.4, respectively.

5.7 Scenario 3: Zero-Day Attacks Detection

In this section, the IDS model relies on the similarity-based learning of the Siamese network to detect zero-day attacks. The experiment evaluates how accurate the similarity measure can detect attacks that are dissimilar to all classes involved during the training process, i.e. zero-day attacks. Zero-Day attacks are flagged when their similarity, with all known classes, is below a certain threshold. The threshold is decided based on the model optimisation and training.

The distinction between One-Shot (Section 5.6) and zero-day detection presented in this section is that, in the One-Shot scenario, newly detected attacks have a few labelled instances that are not sufficient for retraining, while in a zero-day detection scenario, there are no available data for the new attack. Therefore, the model is utilised to detect instances that do not match any of the known classes. After the instances are flagged as unknown, they can be filtered and labelled with the help of other methods (experts for example) and then can be used for the learning of IDS.

5.7.1 Methodology

Figure 5.10 shows the process of building the intrusion detection model and how its similarity learning approach is applied to detect unknown attacks (i.e. zero-day attacks). The process is similar to Figure 5.9, however, after excluding class e, it is not assumed that any of its labelled instances are available, thus is not used in the testing pool of instances. During the evaluation process, a similarity threshold is used. If the

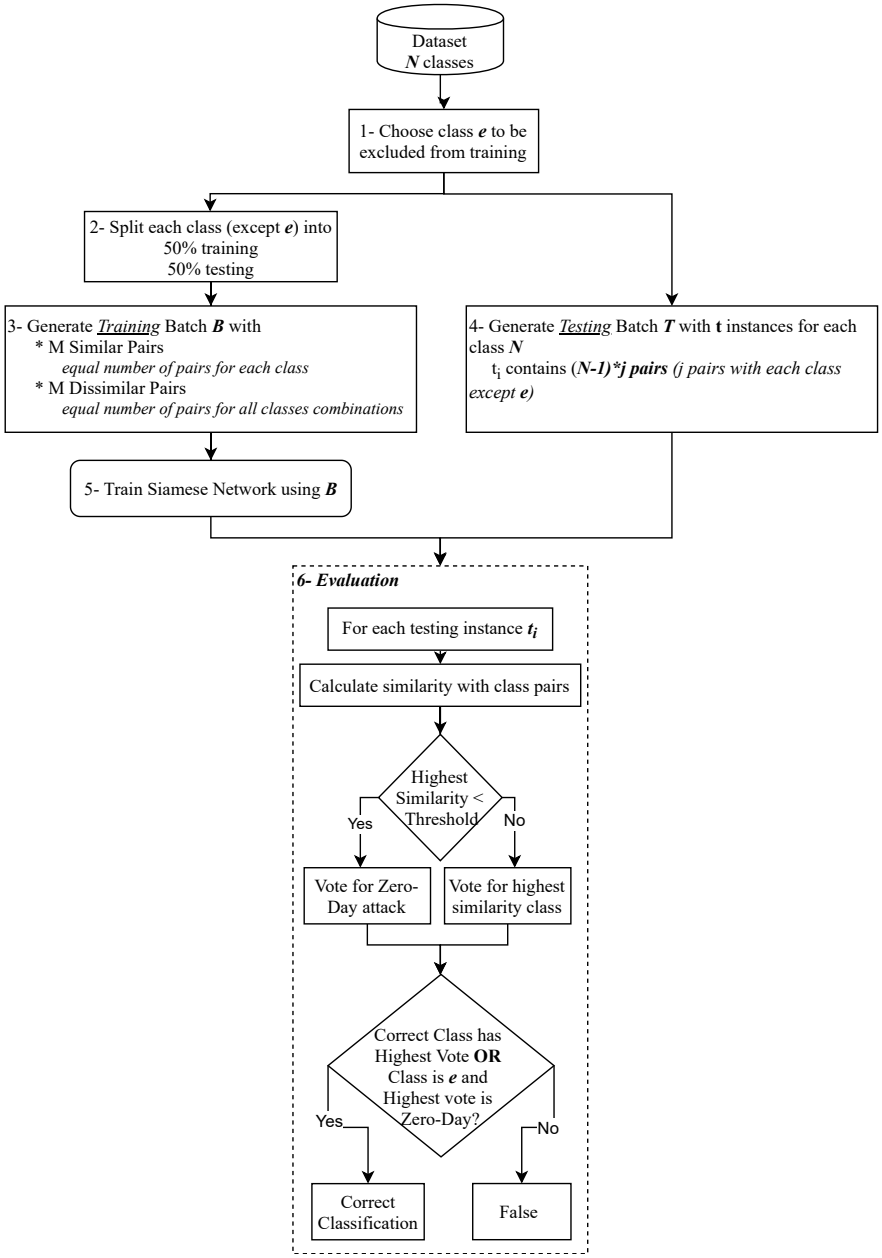


Figure 5.10
Siamese Network for Intrusion Detection (Zero-Day Detection)

similarity is larger than the threshold, then the instance is suspected to be a zero-day attack (knowing that the similarity/distance output is in the range [0 - 1]; the closer the value to 0, the more similar the pair is and dissimilar pairs are closer to 1). The determined threshold is stated for each dataset accordingly in the following sections.

The overall algorithm is the same as Algorithm 5.4 except that a different evaluation function is called. The evaluation function is described in Algorithm 5.6.

Algorithm 5.6 Siamese Network: Evaluate Zero-Day

Input: Trained Siamese Network, Batch Size, Excluded Class (e), Threshold (th)
Output: Zero-Day Detection Accuracy

function EVALZERODAYDETECTION(batch_size)
    $n\_correct \leftarrow 0$
    $num\_per\_class \leftarrow batch\_size/N$
    $K \leftarrow N - e$
    for c in N **do**
        for $i = 0$ to num_per_class **do**
            $ins_1 \leftarrow$ random instance $\in c$
            for $j = 0$ to 5 **do**
                $pairs \leftarrow (ins_1,$ random instance $\in K)$
                $similarities \leftarrow model.predict(pairs)$
                if $similarities[argmin(similarities)] < th$ **then**
                    $votes[zero\_day] += 1$
                else
                    $votes[argmin(similarities)] += 1$
                end if
            end for
            if $argmax(votes) == c$ OR $c == e$ AND $argmax(votes) == zero\_day$ **then**
                $n\_correct += n\_correct + 1$
            end if
        end for
    end for
    $accuracy = n\_correct * 100/batch\_size$
    return $accuracy$
end function

5.7.2 Experiments and Results

5.7.2.1 SCADA Dataset Results

Based on the limitations regarding the SCADA dataset that was discussed in the classification results (Section 5.5.2.1) and the One-Shot results (Section 5.6.2.1), similarity-based zero-day detection is not anticipated to develop high detection accuracies. However, for the completeness benefit, the SCADA results are outlined in this section.

For instance, Table 5.28 and Table 5.29 present the results when person hitting with high intensity and DoS classes are used to mimic zero-day attacks, respectively. Each table lists the overall accuracy using different number of pairs for voting alongside the percentage of attack instances flagged as unknown (i.e. zero-day correctly classified) and as benign (i.e. zero-day attack classified as normal behaviour). Then, the tables show the same for the benign class instances, the percentage of classification as unknown and as benign. Finally, the percentage of known attacks (the ones used during training) that are classified as zero-day is listed.

Table 5.28
Siamese Network: SCADA Zero-Day Accuracy (Person Hitting High Intensity excluded from Training) Using Different j Votes

No Votes	Overall Accuracy	Zero-Day Class (S8)		Benign Class		Known Attack Classes
	Classified As:					
(j)		Unknown	Normal	Normal	Unknown	Unknown
1	61.69%	50.8%	22.3%	23.65%	25.9%	10.59%
5	64.46%	44.1%	33.35%	33.1%	22.1%	9.25%
10	64.38%	64.5%	22.1%	20.45%	47.5%	13.48%
15	65.49%	55.65%	30.55%	26.05%	35.9%	11.05%
20	65.25%	67.45%	22.25%	19.3%	49.7%	13.11%
25	65.1%	74.05%	17.8%	13.65%	58.25%	14.57%
30	65.57%	69.85%	21.3%	17.7%	51.15%	13.1%

Table 5.29
Siamese Network: SCADA Zero-Day Accuracy (DoS excluded from Training) Using Different j Votes

No Votes	Overall Accuracy	Zero-Day Class (S4)		Benign Class		Known Attack Classes
	Classified As:					
(j)		Unknown	Normal	Normal	Unknown	Unknown
1	76.64%	100%	0%	40.15%	5.6%	4.75%
5	79.32%	100%	0%	63.3%	3.05%	4.15%
10	78.8%	100%	0%	60.65%	16%	6.55%
15	80.59%	100%	0%	71.75%	6.35%	4.8%
20	79.99%	100%	0%	70.65%	11.4%	5.91%
25	79.72%	100%	0%	69.45%	16.65%	6.77%
30	80.6%	100%	0%	75.2%	10.2%	5.63%

The overall accuracy falls in the same range as the one reported in the classification and One-Shot sections, which reach 65.57% and 80.6%. Furthermore, it is observed in Table 5.28 and Table 5.29 that the zero-day class detection accuracies reached 74.05% and 100% for S8 and S4, respectively. Also, the percentage of attacks detected as benign is around 20% and 0%, respectively and the percentage of known attacks detected as zero-day attacks reach a maximum of 14.57% and 6.77%, respective. This is crucial as it conveys these attacks were misclassified as other attack classes which ensures detection and the chance for taking corrective and mitigation actions. The rest of the result tables are listed in Appendix E.1. The results show that the similarity, in this case, does not flag zero-day attacks effectively. This is not only because of the classes overlap problem, but also due to the scarcity of features in the SCADA dataset. The dataset provides the sensors recordings only [81].

5.7.2.2 CICIDS2017 Dataset Results

The CICIDS2017 dataset encounters the highest performance in terms of zero-day detection. When excluding SSH brute-force attack class from training and using it to mimic zero-day attacks, 84.8% of the new class instances are correctly detected as unknown. The overall accuracy reached 82.4% as demonstrated in Table 5.30. Moreover, 9.85% of the known attacks are detected as unknown.

Similarly, when DoS (Hulk) class is used to mimic a zero-day attack, 94.17% of the zero-day attack is detected as unknown with an overall accuracy of 76.14%. The performance is outlined in Table 5.31. For completeness, the CICIDS2017 result tables when other attack classes mimic zero-day attacks are listed in Appendix E.2. FTP class experiences a low zero-day detection accuracy and the instances are misclassified as other attack classes, with a very low classification of [3-5]% as normal. DoS (Slowloros) class, on the other hand, reaches a zero-day detection accuracy of 88.88%.

Table 5.30
Siamese Network: CICIDS2017 Zero-Day Accuracy (SSH excluded from Training) Using Different j Votes

No Votes	Overall Accuracy	Zero-Day Class (SSH)		Benign Class		Known Attack Classes
	Classified As:					
(j)		Unknown	Normal	Normal	Unknown	Unknown
1	65.98%	65.28%	18.13%	44.73%	44.83%	20.74%
5	77.74%	65.75%	21.72%	67.52%	28.17%	7.97%
10	80.35%	82.23%	11.1%	64.82%	33.67%	9.91%
15	81.48%	79.57%	13.82%	72.95%	25.6%	8.86%
20	81.87%	84.3%	9.93%	70.65%	28.37%	9.67%
25	81.92%	85.92%	8.45%	69.45%	29.67%	10.06%
30	82.44%	84.8%	9.58%	73.13%	25.98%	9.58%

Table 5.31
Siamese Network: CICIDS2017 Zero-Day Accuracy (DoS (Hulk) excluded from Training) Using Different j Votes

No Votes	Overall Accuracy	Zero-Day Class (DoS (Hulk))		Benign Class		Known Attack Classes
	Classified As:					
(j)		Unknown	Normal	Normal	Unknown	Unknown
1	57.47%	82.28%	15.88%	38.97%	50.85%	35.14%
5	71.53%	78.83%	19.82%	61.37%	33.37%	17.8%
10	73.13%	90.43%	8.57%	57.85%	39.5%	20.66%
15	74.83%	87.42%	11.53%	67.62%	29.77%	19.3%
20	75.28%	93.07%	5.95%	64.85%	32.83%	20.26%
25	75.48%	95.4%	3.62%	63.72%	34.12%	20.73%
30	76.14%	94.17%	4.82%	68.12%	29.72%	20.27%

5.7.2.3 KDD Cup'99 and NSL-KDD Datasets Results

The zero-day attack detection results of the KDD Cup'99 and NSL-KDD datasets when R2L is excluded from training are presented in Table 5.32 and Table 5.33, respectively.

As shown, the overall accuracy reaches 72.98% and 71.04% using 30 pairs in voting. More importantly, the 85.85%, and 72.83% of the R2L class (zero-day class) are correctly flagged as unknown. Around 25-30% of the known attack instances are classified as unknown attacks. The rest of the classes zero-day detection result tables are presented in Appendix E.3 and Appendix E.4.

Table 5.32
Siamese Network: KDD Cup'99 Zero-Day Accuracy (R2L excluded from Training) Using Different j Votes

No Votes	Overall Accuracy	Zero-Day Class (R2L)		Benign Class		Known Attack Classes
	Classified As:					
(j)		Unknown	Normal	Normal	Unknown	Unknown
1	58.1%	87.98%	8.6%	44.4%	46.9%	42.36%
5	71.65%	86.03%	10.1%	62.27%	27.83%	25.67%
10	72.39%	86.27%	9.9%	63.7%	26.5%	25.38%
15	73%	85.92%	10.22%	64.93%	25.02%	24.52%
20	72.97%	85.88%	10.25%	65.07%	24.98%	24.62%
25	72.97%	85.88%	10.25%	65.13%	24.9%	24.64%
30	72.98%	85.85%	10.28%	65.15%	24.87%	24.62%

Table 5.33
Siamese Network: NSL-KDD Zero-Day Accuracy (R2L excluded from Training) Using Different j Votes

No Votes	Overall Accuracy	Zero-Day Class (R2L)		Benign Class		Known Attack Classes
	Classified As:					
(j)		Unknown	Normal	Normal	Unknown	Unknown
1	56.7%	78.73%	14.42%	62.37%	36.1%	51.33%
5	69.44%	72.87%	18.03%	79.18%	20.38%	33.69%
10	69.62%	73.1%	18.12%	79.72%	20%	33.65%
15	70.87%	72.72%	18.15%	79.82%	19.88%	31.46%
20	70.82%	72.85%	18.13%	79.77%	19.95%	31.58%
25	70.81%	72.85%	18.12%	79.75%	19.97%	31.58%
30	71.04%	72.83%	18.13%	79.77%	19.93%	31.19%

5.8 Summary

In this chapter, a novel IDS implementation that leverages One-Shot learning was presented. The models were built using Siamese networks. The objective is to build models that can learn using a limited number of instances. To achieve this goal, three usage scenarios were proposed and evaluated using four datasets. The datasets covered CI networks and general-purpose networks as well.

In the first usage scenario, a Siamese network is trained to classify attacks using limited instances during training. The aim is to evaluate to applicability of similarity-based learning for cybersecurity use. ANN were used as the building block of the Siamese twin networks and random search hyperparameter optimisation alongside the literature hyperparameters values recommendations [277, 276] are performed and loss curves are monitored to ensure the network convergence. The results demonstrated that similarity-based learning using Siamese networks is indeed applicable for cybersecurity use. However, a trade-off was encountered between the number of overlapping classes and the effectiveness of the similarity-based learning. This was demonstrated in the performance of the SCADA dataset, where some classes classification accuracy reached 100%, while others were below 50%. The CICIDS2017 dataset experienced the highest classification accuracy that reached 84.771% with a FPR of 6.15%.

In the second usage scenario, the Siamese network is trained using $N - 1$ classes. After training, a class was added to the network without retraining. The new class represents the case where a new attack is identified and a few labelled instances are available to represent it, however, the instances are not enough to train IDS model. In this case, the Siamese network model is evaluated on its adaptability to correctly classify known attacks (the ones that were used during training) and a new attack. The classification accuracy of attacks that were excluded from the training process demonstrated the applicability of this approach. For the CICIDS2017 dataset, The

SSH Brute-force classification, when mimicking a new attack, reached 78.45% while the FTP Brute-force reached 78.48%. The NSL-KDD and the KDD Cup'99 datasets results confirmed the significance of having a few, yet representable, instances to represent the new cyber attack class. This can be observed in the DoS classification accuracy that rose from 39.88% to 85.37% for the NSL-KDD dataset.

Finally, in the third usage scenario, the Siamese network is further utilised by leveraging the similarity to detect zero-day attacks. In this case, the new cyber attack is assumed to be unknown and no instances exists to represent it. The similarity-based comparison then discriminates instances that fall out of the accepted similarity threshold. The Siamese network was capable of discriminating 84.8% of the SSH and 94.17% of the DoS (Hulk) attacks in the CICIDS2017.

Overall, the experiments and results demonstrate the ability of the proposed Siamese network model to classify cyber attacks based on learning from similarity. Furthermore, the results show the ability of the model to adapt to new cyber attacks and zero-day attacks without the need for retraining. The code is available on GitHub at https://github.com/AbertayMachineLearningGroup/siamese-network-for-IDS.

Chapter 6

Outlier-Based Zero-Day Attacks Detection

6.1 Problem Statement

Detecting zero-day cyber attacks is a challenging task due to their complexity and the pace at which they evolve and emerge [295]. Current ML-based IDS achieve high detection accuracy for known attacks, but they are less effective at detecting unknown zero-day attacks. This is due to the limitations of the models employed by current IDS. With the advancement of ML and DL in domains like image and video processing, Natural Language Processing (NLP), etc., researchers started to leverage these techniques for cybersecurity usage. Nguyen and Reddi [296] discuss the importance and benefit ML can bring to cybersecurity by granting a "robust resistance" against attacks.

As defined by Chapman, a zero-day attack is "a traffic pattern of interest that in general has no matching patterns in malware or attack detection elements in the network" [297]. Bilge and Dumitras [298] discuss the implications of zero-day attacks in the real world focusing on their impact and prevalence. The authors highlight

that zero-day attacks are significantly more prevalent than suspected, demonstrating that out of the 18 attacks they analysed, 11 (61%) were previously unknown [298]. Furthermore, based on the authors' findings, a zero-day attack can exist for a substantial period of time, with an average of 10 months [298], before being detected, thus compromising the target system during that period. The number of zero-day attacks encountered in 2019 exceeds the previously reported figures of the last three years [299]. As a result of all these discussed dimensions, there is a need for an effective detection for zero-day attacks.

In Chapter 5, Siamese networks were utilised to detect zero-day attacks alongside classifying known attacks. The Siamese network was used to flag instances that are dissimilar to all known classes (benign and known attacks) as zero-day attacks.

Recent research uses outlier-based techniques to detect zero-day attacks (i.e., instances/occurrences that vary from benign traffic). However, the main drawback of current outlier-based IDS research is that they have relatively low accuracy rates as a result of both high FPR and high FNR [300]. The high FNR leaves the system vulnerable to cyber attacks [301] and the high FPR needlessly consumes the time of cybersecurity operation centres, leading to "alert fatigue" or "cybersecurity fatigue" [302]. This is evidenced in a study by Cisco that shows that only 28% of the investigated intrusions are real [301]. Therefore, this limits the performance and practicality of deploying the models in real-life.

This chapter focuses on building a model that is capable of detecting zero-day attacks efficiently. The aim is to build models with high detection rates while keeping the false-negatives to a minimum. The proposed methodology leverages the encoding-decoding of autoencoders, which benefits from their training technique that minimises the reconstruction error. By training using benign traffic only, the model can flag unknown attacks. To further demonstrate the efficiency of the autoencoder model, besides comparing the results with recent research, the results are compared against

a One-Class SVM. One-Class SVM is considered one of the robust novelty detection models and has proven its effectiveness and high accuracy in the literature [303].

6.2 Background

The two models that are utilised in this part of the research are explained in this section. Autoencoder is outlined in Section 6.2.1, while Section 6.2.2 discusses the unsupervised variant of a SVM (One-Class SVM) model. Finally, Section 6.2.3 provides an overview of recent IDS research that uses autoencoders.

6.2.1 Autoencoders

The zero-day detection model that is presented in this chapter benefits from autoencoder characteristics and attributes, specifically the encoding-decoding capabilities. The objective is that the autoencoder acts as an outlier-based zero-day attack detector. In this case, the autoencoder model is used to perform binary classification (i.e., benign and zero-day attack) and not multi-class classification.

Rumelhart *et al.* [304] are the first to introduce autoencoders. Their aim is to overcome the back propagation in an unsupervised context by using the input as the target. Autoencoders are categorised as self-supervised learners since the input and output are the same, and the model performs representation learning [305]. As defined by Goodfellow *et al.* [5], an autoencoder is *"a neural network that is trained to attempt to copy its input to its output"* [5].

The basic architecture of an autoencoder is represented in Figure 6.1. The architecture of an autoencoder and the number of hidden layers differ based on the domain and the usage scenario [306]. Formally, given an input vector $\mathcal{X}$, where X represents the feature vector and its size is determined based on the number of features in a dataset, an autoencoder is trained to minimise the reconstruction error, which is

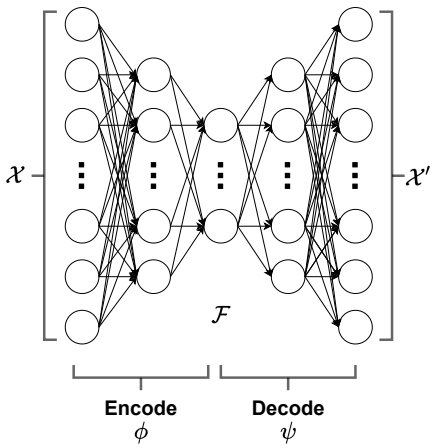


Figure 6.1
Autoencoder Architecture

represented in Equation 6.1 [305], such that ϕ and ψ are the encoding and decoding functions, respectively.

$$\begin{aligned} \boldsymbol{\phi} &: \boldsymbol{\mathcal{X}} \rightarrow \boldsymbol{\mathcal{F}} \\ \boldsymbol{\psi} &: \boldsymbol{\mathcal{F}} \rightarrow \boldsymbol{\mathcal{X}'} \\ \boldsymbol{\phi}, \boldsymbol{\psi} &= \underset{\boldsymbol{\phi},\boldsymbol{\psi}}{\mathbf{argmin}} \, ||\boldsymbol{\mathcal{X}} - (\boldsymbol{\phi} \circ \boldsymbol{\psi})\boldsymbol{\mathcal{X}}||^{\mathbf{2}} \end{aligned} \tag{6.1}$$

Commonly, the reconstruction error of an input $\mathcal{X}$ is represented as the difference between $\mathcal{X}$ and $\mathcal{X}'$. $\mathcal{X}' = g(f(\mathcal{X}))$, where $f(x)$ is the encoding function ϕ, which constructs the encoded vector of $\mathcal{X}$. $g(x)$ is the decoding function ψ, which reconstructs/restores the encoded vector of $\mathcal{X}$. Mean square error (L2 norm) and mean absolute error (L1 norm) are common functions that are used to calculate the reconstruction error as shown in Equation 6.2 and Equation 6.3, respectively, where n is the number of features (data points).

$$||x||_2 = \sqrt{\sum_{i=1}^{n}(x_i - x_i')^2} \tag{6.2}$$

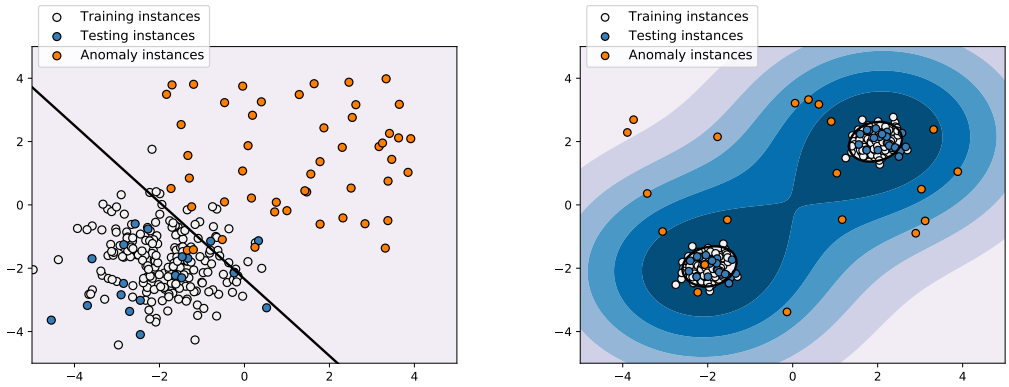
$$||x||_1 = \sum_{i=1}^{n}(|x_i - x_i'|) \tag{6.3}$$

Autoencoders are popular for dimensionality reduction and feature learning [307, 308]. An autoencoder can be seen similar to Principal Component Analysis (PCA) when its encoding function $f(x)$ is a single-layer network with a linear function [309]. In this case the autoencoder adds neither non-linearity to the output nor depth (one layer), which is what PCA does by learning linear transformation of features to another space. This results in a similar output feature space [310]. However, various other applications have been recently proposed for autoencoders in the literature including: word semantics [311], image compression [312], image anomaly detection [313], and denoising [314].

6.2.2 One-Class SVM

SVM is one of the well-established supervised ML techniques. Unlike supervised SVM, One-Class SVM is an unsupervised variant. It is defined as a model capable of detecting "*Novelty*" [315], first proposed by Schölkopf *et al.* [316]. The training goal of One-Class SVM is to fit a hyperplane that acts as a boundary which best comprises all the training data and excludes any other data points. The result of training a One-Class SVM can be visualised as a spherically shaped boundary [317]. Since One-Class SVM is considered one of the most established outlier-based ML techniques, it provides an ideal comparison for assessing the performance of the proposed autoencoder.

Formally, given a class of instances $\{x_1, ..., x_N\}$, and a mapping function $\varphi()$ that maps the features to a space H, the goal of One-Class SVM is to "fit a hyperplane Π in H that has the largest distance to the origin, and all $\varphi(x_i)$ lie at the opposite side of hyperplane to the origin" [318]. Figure 6.2a and Figure 6.2b show examples of the One-Class SVM boundary when using linear and RBF kernels, respectively.



(a) Linear Kernel **(b)** RBF Kernel [319]

Figure 6.2
One-Class SVM Boundaries Example

6.2.3 Related Work

Autoencoders have been proposed for cybersecurity usage for feature engineering and learning. For example, in the work by Kunang *et al.* [320], autoencoders are used for feature extraction, then the features are used in a multi-class SVM classifier. The authors use KDD Cup'99 and NSL-KDD datasets for evaluation. The evaluation results of the model are an overall accuracy of 86.96% and a precision of 88.65%. The different classes accuracies show a highly varying performance as follows; 97.91%, 88.07%, 12.78%, 8.12%, and 97.47% for DoS, probe, R2L, U2R and normal, respectively, a precision of 99.45%, 78.12%, 97.57%, 50% and 81.59% for DoS, probe, R2L, U2R and normal, respectively.

Kherlenchimeg and Nakaya [321] use a sparse autoencoder to extract features. The bottleneck layer of the autoencoder (latent representation) is used as an input to a Recurrent Neural Network (RNN) classifier. NSL-KDD dataset is used for evaluation to reach an 80% accuracy. In a similar fashion, Shaikh and Shashikala [322] use a stacked autoencoder with an LSTM classifier to detect DoS attacks. Using the NSL-KDD dataset, the overall detection accuracy is 94.3% and a FNR of 5.7%. Abolhasanzadeh [323] uses autoencoders for dimensionality reduction and the

extraction of bottleneck features. The experiments are evaluated using the NSL-KDD dataset. In addition, AL-Hawawreh *et al.* [167] train deep autoencoders on benign traffic to deduce the most important feature representation to be used in their deep feed-forward ANN. Shone *et al.* [168] use a stacked Non-Symmetric Deep autoencoder to refine and learn the complex relationships between features. The authors use RF for classification and evaluate their model using both KDD Cup'99 and NSL-KDD datasets. Farahnakian and Heikkonen [324] use a deep autoencoder where it is fed into a single supervised layer for classification. The KDD Cup'99 dataset is used and the highest accuracies are 96.53% and 94.71% for binary and multi-class classification, respectively. In all these experiments, autoencoders are used alongside other models that perform the classification task.

6.3 Datasets

Three mainstream IDS datasets are chosen to evaluate the models proposed in this chapter. CICIDS2017, NSL-KDD, and KDD Cup'99 are the datasets used for evaluation.

The CICIDS2017 dataset [63] covers a wide range of recent insider and outsider attacks in a 5-day recording. It contains a diverse coverage of protocols and attack variations and it is provided in a raw format which allows the flexibility of processing the dataset. Table 6.1 summarises the traffic scenarios recorded per day. The raw files of the CICIDS2017 dataset are preprocessed as discussed below. The full CICIDS2017 description is available in [325].

The second and third datasets are the NSL-KDD [77] and the KDD Cup'99 [78]. Both datasets cover normal traffic and 4 cyber attack classes, namely, DoS, Probing, R2L, and U2R, and are provided in CSV feature files. Each instance is represented with its feature values alongside the class label. The feature files are prepared for ML usage by undergoing categorical feature encoding.

Table 6.1
CICIDS2017 Attacks

Day	Traffic
Monday	Benign
Tuesday	SSH and FTP Brute-force
Wednesday	DoS/DDoS and Heartbleed
Thursday	Web Attack (Brute-force, XSS, Sql Injection) and Infiltration
Friday	Botnet, Portscan and DDoS

Benign traffic instances are solely used to train the models. The benign instances are split into training and validation [326] using sklearn "train_test_split" function [326]. Each of the attack classes is then used to mimic a zero-day attack, thus assessing the ability of the model to detect it. Since the NSL-KDD dataset is provided in two files;"KDDTrain+.csv" and "KDDTest+.csv", attacks in both files are used for evaluation.

6.3.1 CICIDS2017 Dataset Preprocessing

To prepare the CICIDS2017 dataset, the process is outlined as follows. Firstly, the PCAP files of the CICIDS2017 dataset are split based on the attack type and the timestamps provided by the dataset owner. As a result, a PCAP file for each attack class is created. Secondly, bidirectional flows features are extracted. It is important to note that flow-based features are better suited for modern IDS development [53]. This is due to the advancement and complexity of networks and the dependence on encrypted traffic. Flow-based features are applicable for both encrypted and unencrypted traffic analysis [53], because their extraction relies on the communication between two nodes, rather than specific packet data. Thirdly, features with high correlation are dropped to minimise model instability [327].

The process of dropping highly correlated features is described in Algorithm 6.1. A threshold of 0.9 is used [328]. Features with correlation less than the threshold are used

for training. This is because features with high correlation have similar impact on the output (i.e., the dependent variable) [328]. Therefore, one of them is dropped. Finally, the features are scaled using a Standard Scaler. This is done to normalise the features to a mean μ of 0 and standard deviation σ of 1, which accelerates the overall training process [40]. It is important to note that only benign instances are used in selecting the features and scaling to ensure zero influence of the cyber attack instances.

Algorithm 6.1 Drop Correlated Features

Input: Benign Data 2D Array, N, Correlation Threshold
Output: Benign Data 2D Array, Dropped Columns

1: $correlation\_matrix \leftarrow data.corr().abs()$
2: $upper\_matrix \leftarrow correlation\_matrix[i,j] \qquad \{i,j \in N : i <= j\}$
3: $dropped \leftarrow i\{i \in N : correlation\_matrix[i,^*] > threshold\}$
4: $data \leftarrow data.drop\_columns(dropped)$
5: **return** $data$, $dropped$

6.4 Methodology

6.4.1 Autoencoder-based model

An ANN is used as the building block for the proposed autoencoder. Random search [275] is used for the ANN hyperparameter optimisation. The ANN architecture, number of epochs, and learning rate are decided based on the output of random search. Random search is known to converge faster than grid search to a semi-optimal set of parameters. It has also been proven to be better than grid search when a small number of parameters are needed [329]. Finally, it limits the possibility of overfitted parameters.

Once the hyperparameters are decided, the model training takes place. Algorithm 6.2 outlines the overall training process. First, benign instances are split into 75%:25% for training and validation [231], respectively. The model is initialised using the optimal ANN architecture (number of layers and number of hidden neurons

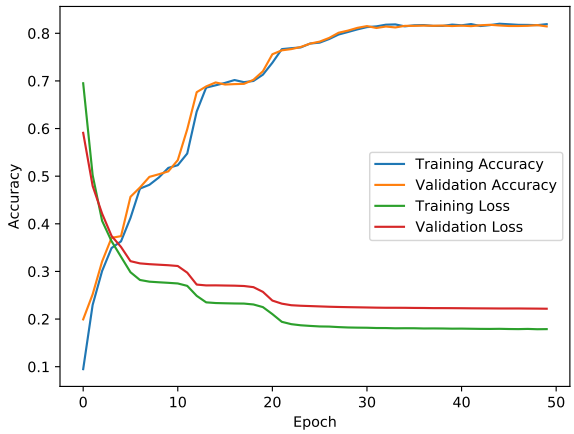


Figure 6.3
Autoencoder Convergence Curve

per layer). Finally, the model is trained for n number of epochs. The loss and accuracy curves are analysed to confirm that the autoencoder converges.

Once the model converges, as shown in Figure 6.3, the model is evaluated using Algorithm 6.3. An attack instance is flagged as a zero-day attack if the Mean Squared Error (MSE) (reconstruction error) of the decoded ($\mathcal{X}'$) and the original instance ($\mathcal{X}$) is larger than a given threshold. The threshold is chosen at first based on the value returned by the random search hyperparameter optimisation. For the purpose of evaluation, multiple thresholds are assessed; 0.05, 0.1, 0.15, 0.2. The threshold plays an important role in deciding the value at which an instance is considered a zero-day attack, i.e., which MSE between $\mathcal{X}'$ and $\mathcal{X}$ is within the acceptable range.

Algorithm 6.2 Autoencoder: Training

Input: benign_data, ANN_architecture, regularisation_value, num_epochs
Output: Trained Autoencoder

1: $training = 75\%$ of $benign\_data$
2: $testing = benign\_data - training$
3: $autoencoder \leftarrow build\_autoencoder$(ANN_Architecture, regularisation_value)
4: $batch\_size \leftarrow 1024$
5: $autoencoder.train(batch\_size, num\_epochs, training, testing)$
6: **return** $autoencoder$

Algorithm 6.3 Autoencoder: Evaluation

Input: Trained Autoencoder, attack, thresholds
Output: Detection accuracies

1: $detection\_accuracies \leftarrow \{\}$
2: $predictions \leftarrow model.predict(attack)$
3: **for** $th \in thresholds$ **do**
4: $\quad accuracy \leftarrow (mse(predictions, attack) > th)/len(attack)$
5: $\quad detection\_accuracies.add(threshold, accuracy)$
6: **end for**
7: **return** $detection\_accuracies$

6.4.2 One-Class SVM based Model

Similar to the autoencoder-based model, One-Class SVM model is trained using the benign instances only. A "ν" value is specified for training the One-Class SVM. Chen *et al.* describe the ν value as "$\nu \in$ [0, 1] which is the lower and upper bound on the number of examples that are support vectors and that lie on the wrong side of the hyperplane, respectively" [330]. The ν default value in scikit-learn library is 0.5 [227]. This means that the goal is to produce in a hyperplane that includes 50% of the training samples. However, for the purpose of this experiment, multiple ν values are chosen (0.2, 0.15, 0.1, 0.05). These ν values were used to evaluate and compare the autoencoder performance.

Algorithm 6.4 shows the process of training the One-Class SVM model. Similar to the autoencoder model in Section 6.2.1, 75% of the benign samples are used in the training process. Unlike the autoencoder model where evaluation relies on a threshold, a trained One-Class SVM model outputs a binary value $\{0,1\}$. The output represents whether an instance belongs to the class to which the One-Class SVM is fit. Hence, each attack is evaluated based on how many instances are predicted with a "0" output of the One-Class SVM.

Algorithm 6.4 One-Class SVM Model

Input: benign_data, nu_value
Output: Trained SVM

1: $training = 75\%\ i \in benign\_data$
2: $testing = benign\_data \cap \overline{training}$
3: $oneclasssvm \leftarrow OneClassSVM(\ nu\_value, `rbf')$
4: $oneclasssvm.fit(training)$
5: **return** $oneclasssvm$

6.5 Experiments and Results

6.5.1 CICIDS2017 Dataset

Autoencoder Results

The optimised architecture for the CICIDS2017 dataset autoencoder model is as follows:

- ANN Architecture: In(18):Hidden(15):Hidden(9):Hidden(15):Out(18)
- Batch size: 1024
- L2 regularisation: 0.0001
- Number of epochs: 50
- Loss: L2 norm (Mean Square Error)

Table 6.2 lists the autoencoder model accuracy of all CICIDS2017 classes. It is important to note that the accuracy is defined differently for benign and attack classes. The model's accuracy reflects the detection of attacks. This relies on the reconstruction error being larger than the given threshold. Unlike attacks, for the benign class the accuracy represents the rate of instances **not** classified as zero-day attacks (i.e., the reconstruction error smaller than the given threshold).

Table 6.2
Zero-Day Detection: CICIDS2017 Autoencoder Results

Class	**Detection Accuracy**			
Threshold	0.2	0.15	0.1	0.05
Benign (Validation)	96.56%	95.19%	90.47%	81.13%
FTP Brute-force	5.18%	5.34%	6.73%	82.82%
SSH Brute-force	7.2%	8.38%	78.05%	80.51%
DoS (Slowloris)	65.63%	71.73%	78.13%	80.85%
DoS (GoldenEye)	66.98%	85.55%	87.71%	90.01%
DoS (Hulk)	98.23%	98.23%	98.34%	98.43%
DoS (SlowHTTPTest)	22.42%	24.03%	28.09%	39.02%
DDoS	83.47%	92.23%	97.88%	99.67%
Heartbleed	28.61%	28.9%	39.6%	43.64%
Web BF	9.7%	9.95%	82.04%	85.41%
Web XSS	11.14%	11.28%	96.38%	99.46%
Web SQL	16.67%	16.67%	22.22%	27.78%
Infiltration - Dropbox 1	47.06%	52.94%	94.12%	94.12%
Infiltration - Dropbox 2	85.71%	85.71%	100%	100%
Infiltration - Dropbox 3	16.3%	23.8%	89.5%	98.04%
Infiltration - Cooldisk	48.08%	51.92%	86.54%	92.31%
Botnet	17.46%	17.77%	37.15%	66.88%
PortScan	16.15%	28.37%	75.21%	98.47%

From Table 6.2, it is noted that the benign class accuracy with a threshold of 0.2, 0.15, 0.1, and 0.05 is 96.56%, 95.19%, 90.47%, and 81.13%, respectively. Furthermore, three categories of attack detection accuracy are observed. Firstly, cyber attack classes that are distinctive from benign which are easily detected. For example, DoS (Hulk) and DDoS where the detection accuracy is high regardless of the threshold [83% - 99%]. Secondly, cyber attack classes that are slightly different from benign (for example, SSH Brute-force, and PortScan). It is observed that the detection accuracy, in this case, depends on the threshold value and an accuracy rise is observed for lower thresholds. This emphasises the influence threshold value choice has on the detection accuracy. Thirdly, cyber attack classes that are not distinguishable from benign traffic,

which are detected but with a lower accuracy (for example, Botnet, SQL Injection and DoS-SlowHTTPTest). These cyber attacks behaviour are similar to the benign traffic behaviour.

Figure 6.4 provides a visualisation of the different CICIDS2017 classes and their corresponding detection accuracies with different threshold values. By observing Figure 6.4, the three discussed categories can be seen, (i) classes with a high stable detection accuracy, (ii) classes with a prompt rise in detection accuracy in the right-most slice (0.05 threshold) and (iii) classes that are not distinguishable from benign traffic. Finally, the benign accuracy (top left) falls within an acceptable range with different thresholds, with 18.87% FPR at most. These can be observed further by plotting the ROC curves for each of the attack classes, as shown in Appendix F.

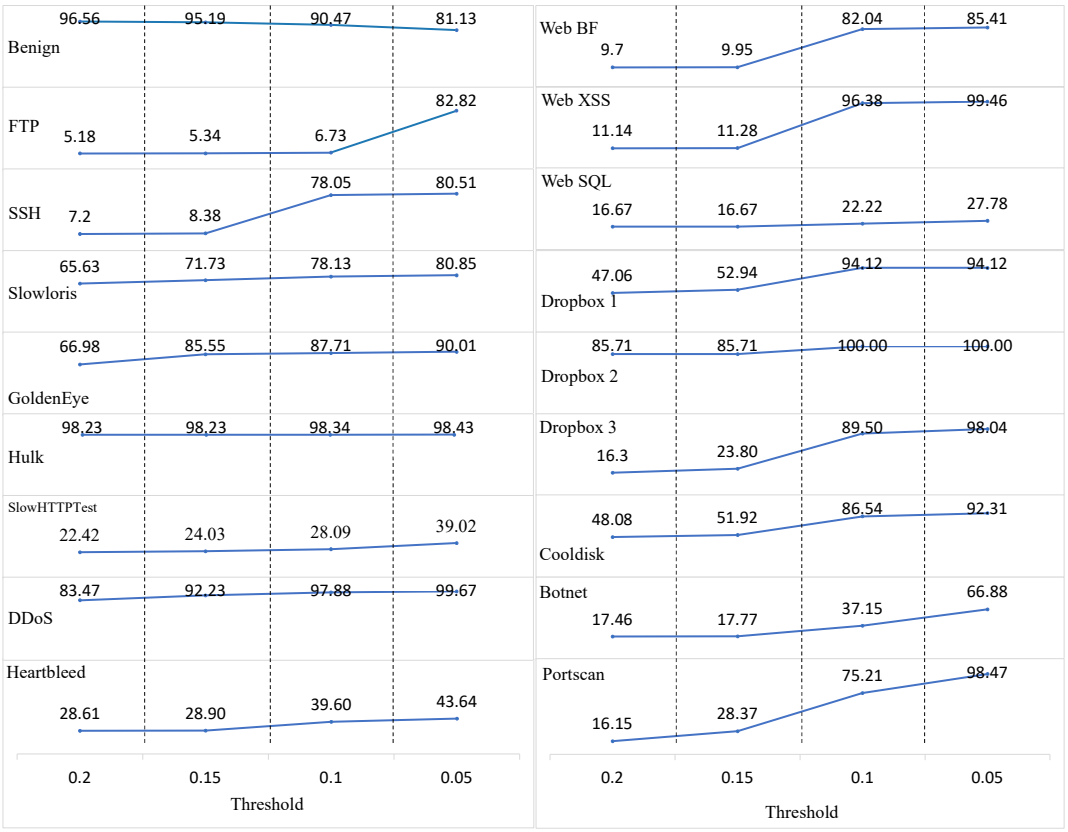


Figure 6.4
CICIDS2017 Autoencoder Detection Results Summary Per Class

One-Class SVM Results

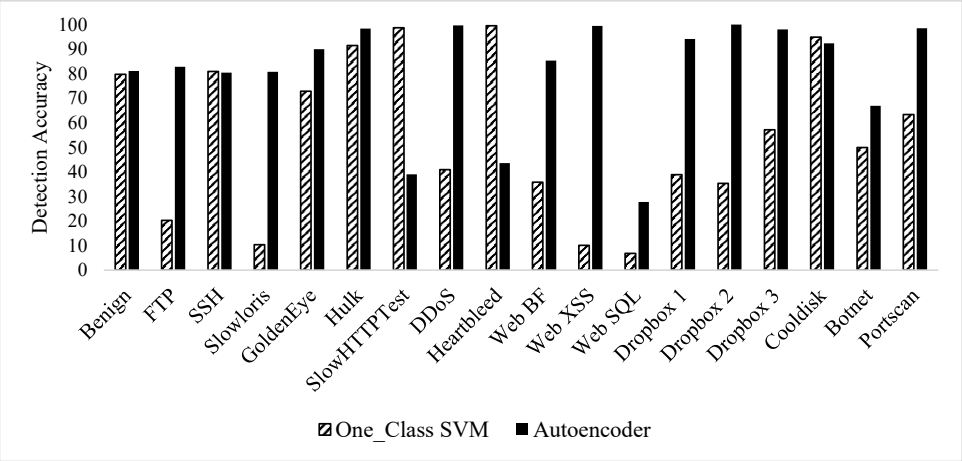
One-Class SVM model results are listed in Table 6.3. By analysing the One-Class SVM results, three observations are identified; (i) The benign detection accuracy decreases when the ν increases. The detection accuracy is 94.84% when $\nu = 0.05$ and 79.71% when $\nu = 0.2$. This is because One-Class SVM model includes more instances within the decision boundary with lower ν. (ii) The classes with high detection accuracy in the autoencoder results (Table 6.2) are detected effectively by the One-Class SVM; however, the One-Class SVM fails to detect the two other categories. This is due to the limitations of the One-Class SVM algorithm, which attempts to fit a hyperplane to separate benign class from other classes. Classes that fall into this hyperplane will always be classified as benign/normal. Finally, (iii) Detection rate of the correctly identified attack classes varies within [0 - 5]% range. For example, DoS (SlowHTTPTest) detection accuracy is 98.11% when $\nu = 0.05$ and 98.71% when $\nu = 0.2$. Similarly, SSH detection accuracy is 78.96% when $\nu = 0.05$ and 80.95% when $\nu = 0.2$.

Table 6.3
Zero-Day Detection: CICIDS2017 One-Class SVM Results

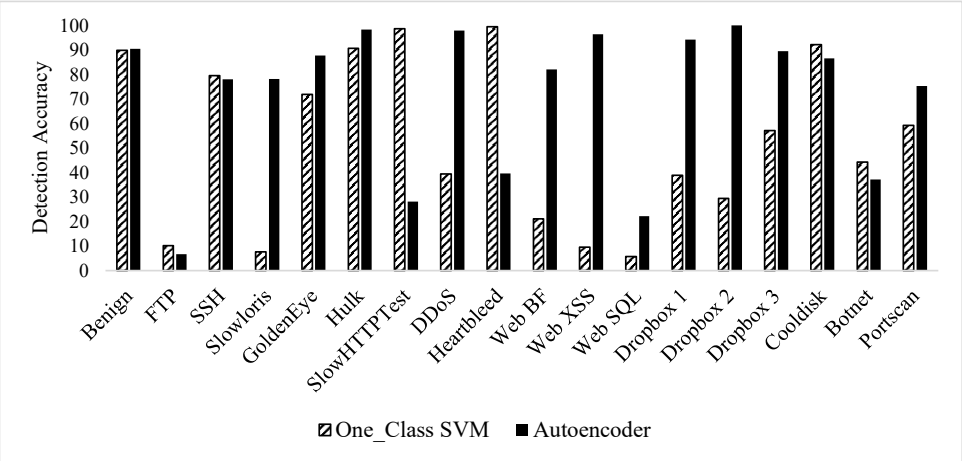
Class	Detection Accuracy			
ν	0.05	0.1	0.15	0.2
Benign (Validation)	94.84%	89.81%	84.84%	79.71%
FTP Brute-force	5.16%	10.19%	15.16%	20.29%
SSH Brute-force	78.96%	79.51%	80.26%	80.95%
DoS (Slowloris)	6.75%	7.66%	8.38%	10.37%
DoS (GoldenEye)	67.32%	71.87%	72.39%	72.85%
DoS (Hulk)	85.73%	90.69%	91.35%	91.55%
DoS (SlowHTTPTest)	98.11%	98.59%	98.66%	98.71%
DDoS	29.89%	39.35%	39.94%	40.96%
Heartbleed	99.09%	99.49%	99.54%	99.58%
Web BF	17.05%	21.1%	23.41%	35.84%
Web XSS	8.38%	9.58%	9.76%	10.13%
Web SQL	5.37%	5.77%	6.31%	6.85%
Infiltration - Dropbox 1	11.11%	38.89%	38.89%	38.89%
Infiltration - Dropbox 2	29.41%	29.41%	35.29%	35.29%
Infiltration - Dropbox 3	57.14%	57.14%	57.14%	57.14%
Infiltration - Cooldisk	90.96%	92.15%	93.8%	94.91%
Botnet	36.54%	44.23%	46.15%	50%
Portscan	57.61%	59.27%	60.04%	63.43%

The comparison of the autoencoder model with the One-Class SVM one is further visualised in Figure 6.5. The two classes that One-Class SVM performs better with than the autoencoder model are DoS (SlowHTTPTest) and Heartbleed. For these two classes the autoencoder reconstruction error was below the zero-day threshold value, however, they were placed on the opposite side of the One-Class SVM hyperplane, which explains their detection accuracy. Therefore, One-Class SVM is well suited for flagging recognisable zero-day attacks. However, autoencoders are better suited for complex zero-day attacks as the performance ranking is significantly higher. Figure 6.5 shows a class-by–by-class comparison of the performance of autoencoder versus One-Class SVM. Figure 6.5 (a) plots the results using One-Class SVM $\nu = 0.2$ and

autoencoder threshold of 0.05, while Figure 6.5 (b) plots the results using One-Class SVM $\nu = 0.09$ and autoencoder threshold of 0.1.



(a) SVM ($\nu = 0.2$), AE (Threshold = 0.05)



(b) SVM ($\nu = 0.1$), AE (Threshold = 0.1)

Figure 6.5
CICIDS2017 Autoencoder and One-Class SVM Comparison

6.5.2 KDD Cup'99 and NSL-KDD Dataset

Autoencoder Results

The autoencoder optimised architecture for the KDD Cup'99 and NSL-KDD datasets is:

- KDD Cup'99 ANN Architecture:
  In(118):Hidden(100):Dr(0.2):Hidden(60):Dr(0.2):Hidden(100):Out(118)
- NSL-KDD ANN Architecture:
  In(118):Hidden(122):Dr(0.2):Hidden(60):Dr(0.2):Hidden(100):Out(122)
- Batch size: 1024
- L2 regularisation: 0.001
- Number of epochs: 50
- Loss: L1 norm (Mean Absolute Error)

It is noted that L1 (Mean Absolute Error) is chosen over L2 for KDD dataset family because it demonstrates better performance. Furthermore, due to the pre-engineered features of these two datasets and their given ranges, L1 provided a better scale for the reconstruction error. Table 6.4 and Table 6.5 list the autoencoder results for the KDD Cup'99 and the NSL-KDD datasets, respectively. Similar to the CICIDS2017 dataset, 75% of the benign class is used for training the autoencoder. For NSL-KDD dataset, attacks in both the KDDTrain+ and KDDTest+ files are used to evaluate the model. As mentioned before, the threshold value is selected based on random search parameter optimisation. The trade-off between the threshold choice and the TNR is observed in the results, however, it is not as significant as the CICIDS2017 dataset discussion. This is due to the limited attack coverage in the KDD datasets and the lower attack discrimination complexity.

Table 6.4
Zero-Day Detection: KDD Cup'99 Autoencoder Results

Class	**Detection Accuracy**		
$Threshold$	0.3	0.25	0.2
Normal (Validation)	87.34%	83.95%	77.64%
DoS	99.4%	99.42%	99.48%
Probe	98.73%	98.93%	99.42%
R2L	96.36%	97.25%	100%
U2R	94.23%	96.15%	98.08%

Table 6.5
Zero-Day Detection: NSL-KDD Autoencoder Results

Class	**Detection Accuracy**		
$Threshold$	0.3	0.25	0.2
KDDTrain+.csv			
Normal (Validation)	79.09%	77.80%	72.78%
DoS	98.15%	98.16%	98.17%
Probe	99.89%	99.94%	99.94%
R2L	83.12%	96.48%	96.48%
U2R	84.62%	100%	100%
KDDTest+.csv			
Normal	84.82%	84.42%	80.94%
DoS	94.67%	94.67%	94.76%
Probe	100%	100%	100%
R2L	95.95%	96.5%	97%
U2R	83.78%	89.19%	100%

Compared to the available autoencoder implementation for detecting zero-day attacks in the literature, the autoencoder results presented in this section outperform [178]. Gharib *et al.* [178] use a hybrid two-stage autoencoder to detect normal and abnormal traffic. Training on KDDTrain+ file and testing on KDDTest+, the authors report an overall accuracy of 90.17%, whereas the proposed autoencoder in this section has the overall accuracy of 91.84%, 92.96% and 94.54% using a threshold of 0.3, 0.25 and 0.2, respectively. Moreover, it is important to note that Gharib *et*

al. [178] do not mention details as of how they define zero-day attacks or the classes they choose in the testing process. Table 6.6 summarises the performance comparison of the autoencoder implementation in this section and the work of Gharib *et al.* [178]. Moreover, it is shown that the implemented autoencoder outperforms the denoising autoencoder proposed in [163]. The authors did not report any use of hyperparameter optimisation or dropping correlated features. Moreover, attack instances influenced their experiments. For example, the authors used the attack instances to train an autoencoder to be able to decide the threshold value they used. Also, features normalisation was performed with attack instances included. The results presented in this work outperforms specifically for the KDDTest+ instances where the authors model's accuracy is capped at 88% while this work reaches 94%.

Table 6.6
Zero-Day Detection: NSL-KDD Performance Comparison

Year	Reference	Approach	Train:Test % of KDDTrain+	KDDTrain+ Accuracy	KDDTest+ Accuracy
This Work		AE th = 0.3 **AE th = 0.25** AE th = 0.2	75 : 25	88.97% **94.48**% 93.47%	91.84% 92.96% **94.54**%
2019	[178]	2 AEs	-	-	90.17%
2017	[163]	AE Denoising AE	80 : 20	93.62% 94.35%	88.28% 88.65%

One-Class SVM Results

For the KDD Cup'99 and NSL-KDD datasets, the One-Class SVM results are reported in Table 6.7 and Table 6.8, respectively. The results show similar detection trends to those of the autoencoder which are discussed in Section 6.5.2. This is due to the limited variance of attacks covered by the KDD Cup'99 and NSL-KDD datasets. To visualise the similarity in detection accuracy, Figure 6.6 and Figure 6.7 show the results for the KDD Cup'99 and NSL-KDD datasets, respectively.

Table 6.7
Zero-Day Detection: KDD Cup'99 One-Class SVM Results

Class	**Detection Accuracy**		
ν	0.1	0.15	0.2
Normal (Validation)	90.15%	85.24%	79.93%
DoS	99.48%	99.49%	99.71%
Probe	99.05%	99.29%	99.37%
R2L	96.8%	97.51%	98.49%
U2R	96.15%	96.15%	98.08%

Table 6.8
Zero-Day Detection: NSL-KDD One-Class SVM Results

Class	**Detection Accuracy**		
ν	0.1	0.15	0.2
KDDTrain+.csv			
Normal (Validation)	89.9%	85.14%	80.54%
DoS	98.13%	98.14%	98.14%
Probe	97.74%	98.77%	99.52%
R2L	49.35%	52.26%	81.71%
U2R	78.85%	80.77%	82.69%
KDDTest+.csv			
Normal	88.12%	86.02%	84.72%
DoS	94.67%	94.67%	94.69%
Probe	99.55%	99.91%	100%
R2L	80.17%	82.22%	90.31%
U2R	78.38%	78.38%	83.78%

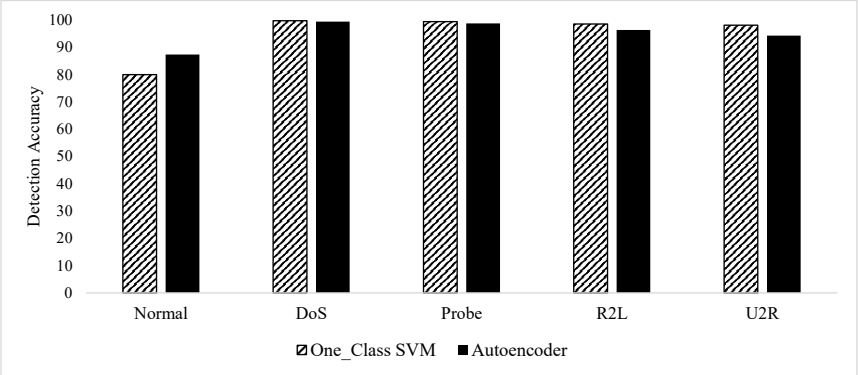


Figure 6.6
KDD Cup'99 Autoencoder and One-Class SVM Comparison
SVM ($\nu = 0.2$), AE (Threshold = 0.3)

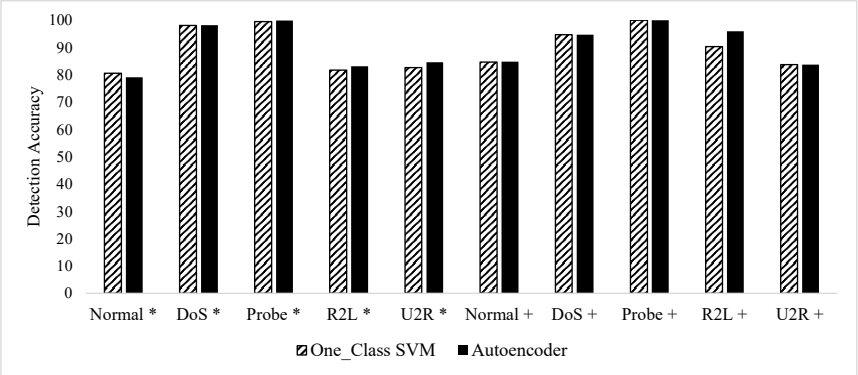


Figure 6.7
NSL-KDD Autoencoder and One-Class SVM Comparison
SVM ($\nu = 0.2$), AE (Threshold = 0.3)
*: KDDTrain+ file, +: KDDTest+ file

6.6 Summary

In this chapter, the zero-day detection problem is tackled from a different prospective. Unlike Chapter 5 where zero-day attacks were detected using a Siamese network trained to classify and discriminate attacks based on similarity, in this chapter, an autoencoder is used. The autoencoder is trained using benign traffic only, then, relying on the encoding-decoding capabilities of the autoencoder, zero-day attacks are detected.

The proposed Autoencoder model is tested using three benchmark datasets, namely, KDD Cup'99, NSL-KDD, and CICIDS2017. The experiments demonstrated a high detection accuracy for zero-day attacks. The CICIDS2017 zero-day detection accuracy reaches 90.01%, 98.43%, 98.47%, and 99.67% for DoS (GoldenEye), DoS (Hulk), PortScan and DDoS attacks, respectively. The KDD Cup'99 dataset detection accuracy reached 95.21% and NSL-KDD dataset detection accuracy reaches 92.96%.

Furthermore, to assess the autoencoder performance, it is compared to an unsupervised outlier-based ML technique; One-Class SVM, which detects outliers. The one-class SVM mode presents its effectiveness in detecting zero-day attacks for KDD Cup'99 and NSL-KDD datasets and the distinctive attack classes from the CICIDS2017 dataset. Compared to One-Class SVM, the autoencoder shows better detection accuracies. Both models demonstrate low FPR. Finally, the CICIDS2017 classes that mimic benign traffic behaviour, DoS (Slowloris), DoS (SlowHTTPTest) as an example, experience lower detection rates by both the autoencoder and the One-Class SVM models. This is due to the tactic used by attackers to ensure that attacks display similar behaviour to benign traffic. This problem - detecting attacks mimicking benign behaviour - is addressed in the next chapter. The code is available on GitHub at https://github.com/AbertayMachineLearningGroup/zero-day-detection.

Chapter 7

Classifying Benign Imitating Attacks Using Flow Aggregation

7.1 Problem Statement

Cyber attacks are becoming more complex due to the expansion of attack surfaces found in hardware and software of modern computing technologies, and the evolution of more advanced evasion methods. As outlined in Chapter 6, there exists attacks that are overlooked in recent research, or - when considered - demonstrate low detection accuracy. One of the reasons behind this low detection accuracy is because these attacks mimic benign traffic behaviour to evade detection mechanisms.

In ML models, the choice of features is more important than the choice of the model [331]. Ghaffarian and Shahriari state that features play a vital role in the development of IDS [42]. The features used in ML-based IDS cover (i) packet-based information; for example, TCP flags, IP Flags, packet length, etc., and (ii) flow-based features that characterise the communication between two nodes; for example, the average size of packets, and average time between packets in a flow [332]. In current research, these features have demonstrated their effectiveness when combined with

feature engineering techniques and sufficient training samples [331, 173]. However, these features are not effective in cases where cyber attacks mimic benign traffic behaviour. This is demonstrated in the difficulty to detect/flag those attacks (i.e. zero-day).

In this chapter, an additional level of feature abstraction, named "*Flow Aggregation*" is proposed to tackle this problem and aid in detecting cyber attacks that mimic benign traffic behaviour. These new features are based on a higher level of abstraction of network traffic. Specifically, flow aggregation is performed by grouping flows initiated from a network host. This additional level of feature abstraction benefits from the cumulative information, thus aiding in qualifying a model to classify benign-mimicking attacks.

The CICIDS2017 dataset is used to evaluate the proposed features with a focus on the attacks that are difficult to detect as shown in Chapter 6; namely, DoS (Slowloris) and DoS (SlowHTTPTest) attacks. The new feature significance is evaluated on attack classes that do and do not mimic benign behaviour. Finally, the experiments that are presented in Chapter 6 are re-evaluated using the proposed features to assess the effect they have on zero-day detection performance.

7.2 Background

7.2.1 Related Work

Different features are used to build IDS. Rezaei and Liu [53] discuss four main categories of networking features, namely; time series, header, payload, and statistical. The authors discuss the advantages of using time series and statistical features in comparison with header and payload features, as the former can be extracted from both encrypted and unencrypted traffic. The authors further highlight the shortcomings of available encrypted traffic classification research [53]. Both packet-based and

flow-based features have been used for intrusion detection purposes and have proved to be effective. However, with the dominance of network encryption, which reached 87% at the beginning of 2019 [333], packet-based features are rendered less reliable at detecting cyber attacks in modern networking.

Older attacks are predominant in datasets like KDD Cup'99, and NSL-KDD. These datasets are used to train ML-based IDS, and in many cases achieve good results. More up-to-date cyber attacks are recorded in the CICIDS2017 dataset [63], therefore, building IDS models using the CICIDS2017 is a more complex undertaking.

Table 7.1 and Table 7.2 provide a list of recent articles in which the CICIDS2017 dataset is used. The tables present the published articles, the ML models applied, the metrics used to assess performance, and the accompanied results. Two observations are noticed in Table 7.1. (i) Research utilising the CICIDS2017 dataset involve a subset of attacks, specifically the ones that are distinctive from benign traffic. DDoS, PortScan, and SSH, for example, have received attention from researchers, whilst others have been overlooked due to their poor results and their benign-like behaviour that render their classification difficult. Studies that include these other attacks demonstrate a low detection accuracy. This low detection accuracy is not reflected in the classification models' overall accuracy due to the class imbalance problem of this dataset [334]. (ii) The overall accuracy is much higher than the accuracy of individual classes. For example, in [176], when 1-layer ANN is used, the overall multi-class classification accuracy is 96% (Table 7.1), while the individual classes detection accuracies are 55.9%, 95.9%, 85.4% and 85.2% for normal, SSH, DDoS and PortScan classes, respectively (Table 7.2). This indicates the misleading effect of reporting the overall accuracy when dealing with imbalanced datasets.

Vinayakumar *et al.* highlight in their recent research on the CICIDS2017 dataset that by observing the saliency map for the dataset, it is shown that "the dataset requires a few more additional features to classify the connection record correctly" [176].

Table 7.1
CICIDS2017 Recent Articles Performance Summary (1)

Year/Ref	Approach	Covered Attacks	Accuracy	Precision	Recall	F-Score
2020/[184]+	MLP	SSH	-	82%	98%	90%
	LSTM		-	97%	98%	97%
	MLP	FTP	-	93%	77%	85%
	LSTM		-	98%	99%	99%
2019/[176]+	DNN (1 Layer)	Binary	96.3%	90.8%	97.3%	93.9%
	DNN (5 Layers)		93.1%	82.7%	97.4%	89.4%
	LR		83.9%	68.5%	85%	75.8%
	NB		31.3%	30%	97.9%	45.9%
	KNN		91.0%	78.1%	96.8%	86.5%
	SVM (RBF)		79.9%	99.2%	32.8%	49.3%
	DNN (1 Layer)	Multi-class	96%	96.9%	96%	96.2%
	DNN (5 Layers)		95.6%	96.2%	95.6%	95.7%
	LR		87%	88.9%	87%	86.8%
	NB		25%	76.7%	25%	18.8%
	KNN		90.9%	94.9%	90.9%	92.2%
	SVM (RBF)		79.9%	75.7%	79.9%	72.3%
2019/[179]	AdaBoost	DDoS	81.83%	81.83%	100%	90.01%
2018/[335]	DL	PortScan	97.80%	99%	99%	99%
	SVM		69.79%	80%	70%	65%
2018/[336]	C5.0	DDoS	85.92%	86.45%	99.70%	-
	RF		86.29%	86.80%	99.63%	-
	NB		90.06%	79.99%	86.03%	-
	SVM		92.44%	79.88%	84.36	-

+ Only snippets of the results are listed in the table.

Where:
DDoS: Distributed Denial of Service
DL: Deep Learning
DNN: Deep Neural Network
FTP: File Transfer Protocol
KNN: k-Nearest Neighbour
LR: Logistic Regression
LSTM: Long short-term memory
MLP: Multilayer Perceptron
NB: Naïve Bayes
RBF: Radial Basis Function
RF: Random Forest
SSH: Secure Shell
SVM: Support Vector Machine

Table 7.2
CICIDS2017 Recent Articles Performance Summary (2)

Year/Ref	Approach	Accuracy			
		Normal	**SSH**	**DDoS**	**PortScan**
2019/[176]+	DNN (1 Layer)	55.9%	95.9%	85.4%	85.2%
	DNN (5 Layers)	56.8%	95.8%	85.5%	85.5%
	LR	88.5%	98.4%	92.2%	92.6%
	NB	32.2%	75.7%	98.5%	87.9%
	KNN	90.9%	97%	99.5%	99.6%
	SVM (RBF)	79.8%	98.8%	92.9%	99%

+ Only snippets of the results are listed in the table.

The authors' observations highlight this need specifically for the DoS class. As later discussed in the experiments and results in Section 7.4, the findings in this chapter concur with this observation regarding the attack classes that need the proposed additional abstraction level of features to be discriminated from benign traffic and other attacks.

7.3 Methodology

Starting with a raw PCAP file which contains network traffic, two levels of features can be extracted as shown in Figure 7.1. The first level (lower level) inspects the individual packets to extract packet-based features. For example, TCP and IP flags, packet size, ports, protocol used, etc. The second level inspects flow to extract flow-based features, either unidirectional or bidirectional. This is done by inspecting all individual packets in a particular communication flow.

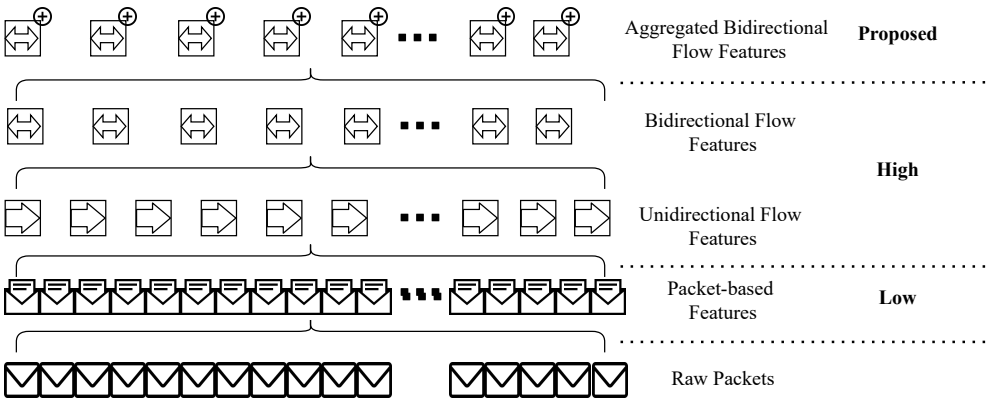


Figure 7.1
Abstraction Levels of Networking Features

The Centre for Applied Internet Data Analysis (CAIDA) defines a flow as "a set of packets which share a common property" [24]. In other words, given two nodes/endpoints in a network, the packets involved in the communication between them are abstracted as network flows. A network flow could be 2-tuple, where the source and destination IP addresses are used. When the source and destination ports are also used, a flow is considered to be 4-tuple, then 5-tuple flows additionally include the protocol used. The 5-tuple flow is the most commonly used one. Network flows can be unidirectional (i.e., host A to host B), or bidirectional, which combines the packets in the unidirectional flow $(A - B)$ and $(B - A)$.

In this chapter, a third additional level of abstraction is proposed. The aim is to represent characteristics and information about the overall communication between hosts. This new level groups bidirectional flows into bundles and aggregated features are derived, called "Flow aggregation features". The features provide additional traffic characteristics in the form of cumulative information.

After these aggregated features are computed, they are propagated back to each bidirectional flow in the bundle/group. This is represented by the superscript + sign in Figure 7.1. The two proposed flow aggregation features in this chapter are (i) number of flows and (ii) source ports delta.

Number of Flows: The first added feature represents the number of siblings in a flow bundle. Given the communication between a host A and one or more hosts, all flows initiated by A are counted. The advantage of this feature is that it is significant for attacks that intentionally spread their associated requests over time when targeting a single host. However, when grouped, the bundled flow will have additional information about how many flows are in the same group that can resemble the communication pattern. Moreover, it can represent patterns when an attacker targets many hosts, each with a few communications, or spread the communication over time. When these flows are grouped, a pattern can be identified.

Figure 7.2 shows how the flows bundling process takes place. Each letter at the top of Figure 7.2 represents a node in the network. Similarly, each pair of arrows in Figure 7.2 represents a bidirectional flow with the notation XY_i, such that X is the source node, Y is the destination node, and i is the communication counter. Finally, the colours in Figure 7.2 represent the grouping of flows into bundles. It is observed that the first bundle (in blue colour) has 4 flows, therefore, AB_1, AB_2, AC_1, and AD_1 will have the "number of flows" feature set to 4. Similarly, the second bundle (in green colour), BC_1 and BC_2 will have the value 2 and so on.

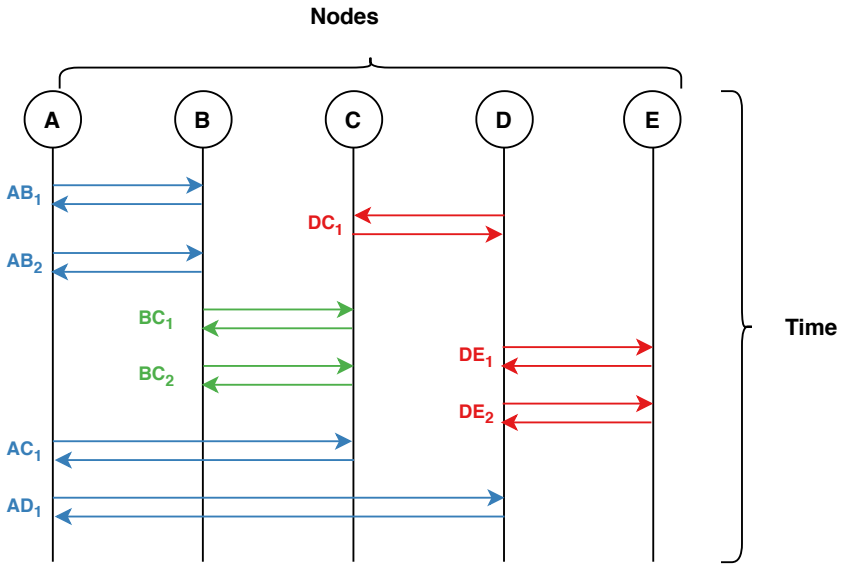


Figure 7.2
Aggregation of Network Traffic Flows. Each Colour Represents an Aggregated Flow

Source Ports Delta: The second added feature is "source ports delta". This feature is calculated using all the port numbers used in a bundle. Algorithm 7.1 illustrates how this feature is calculated. The advantage of this feature is to capture the level and variation pattern of the used ports in legitimate traffic. The feature adds this piece of information to each flow, which then enhances the learning and classification as further discussed in this chapter.

Algorithm 7.1 Flow Aggregation: Calculate Ports Delta Feature

Input: List of bundle flow ports
Output: Ports Delta Feature

```
ports.sort()
for i ∈ length(ports) − 1 do
    diff[i] ← abs(ports[i+1] - ports[i])
end for
avg_diff ← diff.mean()
return avg_diff
```

Recursive Feature Elimination (RFE) [337] is used to validate the significance of the added features when used to classify classes that mimic benign traffic behaviour. RFE is used to select the best k features (here, $k = 5$ [338]). Over the various experiments discussed in Section 7.4, RFE demonstrates that the two features are important for identifying classes that mimic benign behaviour. For attacks that are distinctive, flow aggregation features are nonessential.

The parameters for the classification models are as follows:

- RFE: Logistic Regression with 2000 iterations
- ANN Architecture:
  - Binary Classifier: In(**5**) : 3 : *2*
  - Three-Class Classifier: In(**5**) : 3 : *3*
  - Five-Class Classifier (1): In(**5**) : 3 : *5*
  - Five-Class Classifier (2): In(**10**) : 8 : *5*
- Activation: Relu for hidden layers and Sigmoid for output layer

- Batch size: 64
- Number of epochs: 50
- Optimiser: Adam
- Loss: Mean Square Error

7.4 Experiments Methodology and Results

In this section, different classification experiments are performed to assess the impact of "Flow Aggregation" on different attack classification problems. Moreover, the autoencoder experiment that was evaluated in Chapter 6 is reassessed using the proposed features to examine their significance in zero-day attack detection; specifically, for attacks that were previously detected with low accuracy.

The CICIDS2017 dataset [62] is used for evaluation. The attacks of interest from the CICIDS2017 dataset are DoS (SlowHTTPTest) and DoS (Slowloris). These two attacks implement low-bandwidth DoS attacks in the application layer by draining concurrent connections pool [339]. Since these two attacks are performed slowly, they are hard to detect. Besides DoS (SlowHTTPTest) and DoS (Slowloris), two other attacks are used for comparative purposes; PortScan and DoS (Hulk). These two attacks resemble the case where attacks are easier to discriminate from benign traffic. Since the attacks of interest are underrepresented in the CICIDS2017 dataset [340], a portion of one hour of Monday benign traffic and PortScan are used for the classification purpose [341].

Initially, each of the four attack classes and benign PCAP files are processed to extract features. The output of this process is 5 CSV files containing bidirectional flow features and aggregation features. RFE is then performed to select the best k features which are fed into an ANN classifier. Because the focus is to evaluate the additional level of feature abstraction and not the classifier model complexity, the ANN classifier architecture is straightforward. It is composed of 5 input neurons, 1 hidden layer composed of 3 neurons, and an output layer.

Three classification experiments are performed. The first experiment is a binary classification problem for each of the attacks of interest versus the benign class (Section 7.4.1). The second experiment is a three-class classification (Section 7.4.2). This experiment evaluates the classification of benign, a benign-mimicking attack, and a distinctive attack (i.e., not mimicking benign behaviour). Finally, the third experiment is a five-class classification including all classes of interest (Section 7.4.3). Each of these experiments is performed twice, with bidirectional features only and with bidirectional features and aggregation features. The RFE is performed independently in each experiment and the selected features are listed to highlight the cases where the new features prove significant. For the purpose of performance comparison, the RFE features that are selected without the flow aggregation ones are used alongside the two new features.

7.4.1 Binary Classification Results

This section outlines the results of the first experiment which is a binary classification. Each of the attacks of interest is classified against the benign class. The RFE ranking for the proposed Flow aggregation features in the binary classification (each of the attack classes versus benign class) is outlined in Table 7.3. It can be observed that the new features are in the top list for the benign mimicking attacks (in bold) and not as significant for the distinctive ones.

Table 7.3
Binary Classification Flow Aggregation RFE Ranking

Attack Class	**Flow Aggregation Feature Rank (out of 30 features)**	
	Number of Flows	**Source Ports Delta**
DoS (Slowloris)	10	1
DoS (SlowHTTPTest)	6	5
DoS (Hulk)	5	28
PortScan	1	27

Table 7.4 and Table 7.5 show the precision, recall, and F1-Score for DoS (Slowloris) and DoS (SlowHTTPTest), respectively. The results are calculated using 5-fold cross validations and are written as (Mean ± Standard Deviation). The recall of each of the attack classes rises when the flow aggregation features are included. The recall rises from 83.69% to 91.31% for DoS (Slowloris) attack class and from 65.94% to 70.03% for the DoS (SlowHTTPTest) attack class. Unlike attacks that mimic benign behaviour where flow aggregation features improve the classification performance, classification performance does not hugely benefit from flow aggregation features in the case of distinctive classes (classes that do not mimic benign traffic behaviour). This is observed in Table 7.6 and Table 7.7 for DoS (Hulk) and PortScan classes, respectively. Precision and recall are high for both of these attacks without utilising the aggregation flow features. The increase in precision and recall for DoS (Hulk) is 0.01% and 0.48%, and for PortScan is 0.64% and 0.11%, respectively.

Table 7.4
Benign-DoS (Slowloris) Classification (5-fold cross-validation)

	Without Aggregation			With Aggregation		
RFE Selected Features	1- Fwd Min Inter-arrival Time 2- Bwd Min Inter-arrival Time 3- Bwd mean time between the first packet and each successive packet 4- Fwd mean time between the first packet and each successive packet 5- Fwd STD Inter-arrival Time			Without Aggregation Features + **Number of Flows** + **Source Ports Delta**		
	Precision	Recall	F1-Score	Precision	Recall	F1-Score
Benign	99.04%± 0.08%	99.86%± 0.13%	99.45%± 0.05%	99.49%± 0.08%	99.99%± 0.01%	99.74%± 0.04%
Slowloris	97.35%± 2.35%	83.69%± 1.42%	89.97%± 0.81%	99.73%± 0.26%	91.31%± 1.35%	95.33%± 0.76%

Table 7.5
Benign-DoS (SlowHTTPTest) Classification (5-fold cross-validation)

	Without Aggregation			With Aggregation		
RFE Selected Features	1- Fwd mean time between the first packet and each successive packet 2- Bwd mean time between the first packet and each successive packet 3- Fwd Min Inter-arrival Time 4- Bwd Min Inter-arrival Time 5- Fwd Max Inter-arrival Time			Without Aggregation Features + **Number of Flows** + **Source Ports Delta**		
	Precision	Recall	F1-Score	Precision	Recall	F1-Score
Benign	98.49%± 0.04%	99.94%± 0.02%	99.21%± 0.03%	98.68%± 0.40%	99.87%± 0.14%	99.27%± 0.17%
SlowHTTP Test	98.13%± 0.56%	65.94%± 0.98%	78.87%± 0.82%	96.24%± 3.68%	70.03%± 9.27%	80.63%± 5.21%

Table 7.6
Benign-DoS (Hulk) Classification (5-fold cross-validation)

	Without Aggregation			With Aggregation		
RFE Selected Features	1- Bwd Min Packet Length 2- Fwd Num Reset Flags 3- Bwd Num Push Flags 4- Bwd Num Reset Flags 5- Fwd Max Inter-arrival Time			Without Aggregation Features + **Number of Flows** + **Source Ports Delta**		
	Precision	Recall	F1-Score	Precision	Recall	F1-Score
Benign	99.83%± 0.04%	99.99%± 0.01%	99.91%± 0.02%	100.0%± 0.00%	100.0%± 0.00%	100.0%± 0.00%
Hulk	99.98%± 0.03%	99.51%± 0.10%	99.74%± 0.06%	99.99%± 0.02%	99.99%± 0.02%	99.99%± 0.02%

Table 7.7
Benign-PortScan Classification (5-fold cross-validation)

	Without Aggregation			With Aggregation		
RFE Selected Features	1- Fwd STD Packet Length 2- Bwd Min Packet Length 3- Fwd Max Packet Length 4- Fwd Mean Packet Length 5- Fwd Number of Push Flags			Without Aggregation Features + **Number of Flows** + **Source Ports Delta**		
	Precision	Recall	F1-Score	Precision	Recall	F1-Score
Benign	99.40%± 0.03%	99.36%± 0.93%	99.38%± 0.45%	99.51%± 0.03%	100.0%± 0.00%	99.75%± 0.01%
PortScan	99.36%± 0.92%	99.39%± 0.04%	99.37%± 0.45%	100.0%± 0.00%	99.50%± 0.03%	99.75%± 0.01%

Figure 7.3 shows the effect of using flow aggregation features on the recall of four attack classes. It is observed that the two attack classes that mimic benign behaviour, DoS (SlowHTTPTest) and DoS (Slowloris), experience a rise in recall. However, for the other classes, DoS (Hulk) and PortScan, the recall is high regardless of the use of flow aggregation features. This emphasises the fact that bidirectional flow features are sufficient for benign distinguishable attack classes, but not for the benign-mimicking classes. This observation is further discussed in the following experiments.

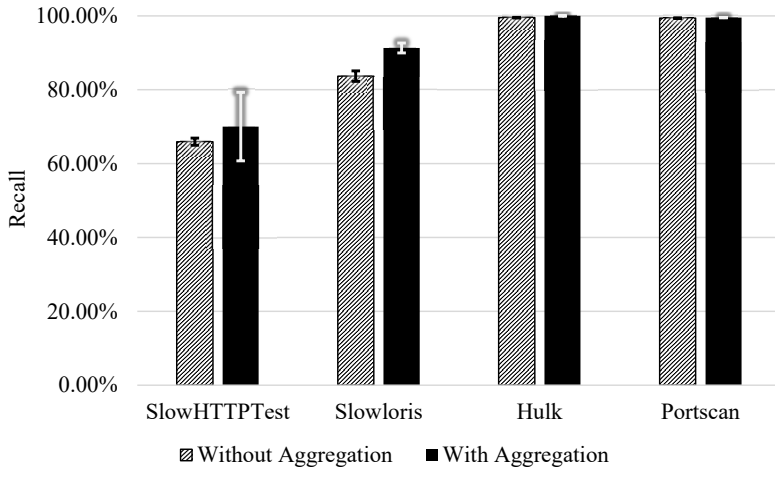


Figure 7.3
Binary Classification — Impact of Flow Aggregation on Classification Recall of Attack Classes (Benign-Attack)

7.4.2 Three-Class Classification Results

In the second experiment, a complexity is added to the binary classification problem by increasing the number of classes. This reduces the likelihood of the benign-mimicking attack to be correctly discriminated. A three-class classification is performed. Benign class alongside a discriminative class that does not mimic benign traffic behaviour (PortScan is used for this purpose) and a benign-mimicking class are used. The RFE ranking for the proposed Flow aggregation features in the three-class classification is listed in Table 7.8.

Table 7.8
Three-Class Classification Flow Aggregation RFE Ranking

Attack Class	**Flow Aggregation Feature Rank (out of 30 features)**	
	Number of Flows	**Source Ports Delta**
DoS (Slowloris)	2	1
DoS (SlowHTTPTest)	1	13
DoS (Hulk)	1	28

The three-class classification results demonstrate similar behaviour to the binary classification ones. The recall of the benign and PortScan classes is high without using the flow aggregation features. However, the recall experiences a high rise in the benign-mimicking attack class.

As shown in Table 7.9, the DoS (Slowloris) recall rises from 78.25% to 99.09%, while in Table 7.10, the DoS (SlowHTTPTest) recall rises from 0% to 58.97%. Finally, the DoS (Hulk) class recall rises from 98.56% to 99.50% as shown in Table 7.11.

Table 7.9
Benign-PortScan-DoS (Slowloris) Classification (5-fold cross-validation)

	Without Aggregation			With Aggregation		
RFE Selected Features	1- Fwd STD Packet Length 2- Bwd Min Packet Length 3- Bwd mean time between the first packet and each successive packet 4- Fwd mean time between the first packet and each successive packet 5- Fwd Max Packet Length			Without Aggregation Features + **Number of Flows** + **Source Ports Delta**		
	Precision	Recall	F1-Score	Precision	Recall	F1-Score
Benign	98.31%± 0.16%	97.69%± 1.02%	98.00%± 0.49%	99.46%± 0.04%	99.99%± 0.01%	99.73%± 0.02%
PortScan	97.85%± 0.99%	99.60%± 0.12%	98.71%± 0.45%	100.0%± 0.01%	99.50%± 0.03%	99.74%± 0.01%
Slowloris	96.95%± 1.62%	78.25%± 1.68%	86.59%± 1.45%	99.75%± 0.15%	99.09%± 0.44%	99.42%± 0.21%

Table 7.10
Benign-PortScan-DoS (SlowHTTPTest) Classification (5-fold cross-validation)

	Without Aggregation			With Aggregation		
RFE Selected Features	1- Fwd Mean Packet Length 2- Fwd STD Packet Length 3- Fwd Max Packet Length 4- Bwd mean time between the first packet and each successive packet 5- Fwd mean time between the first packet and each successive packet			Without Aggregation Features + **Number of Flows** + **Source Ports Delta**		
	Precision	Recall	F1-Score	Precision	Recall	F1-Score
Benign	95.10%± 0.04%	96.40%± 0.12%	95.75%± 0.06%	97.67%± 1.25%	99.99%± 0.01%	98.81%± 0.64%
PortScan	96.45%± 0.11%	99.52%± 0.05%	97.96%± 0.06%	99.99%± 0.02%	99.42%± 0.15%	99.70%± 0.08%
SlowHTTP Test	0.00% ± 0.00%	0.00% ± 0.00%	0.00% ± 0.00%	79.62%± 39.81%	58.97%± 29.88%	67.67%± 34.00%

Table 7.11
Benign-PortScan-DoS (Hulk) Classification (5-fold cross-validation)

	Without Aggregation			With Aggregation		
RFE Selected Features	1- Fwd Mean Packet Length 2- Fwd Max Packet Length 3- Fwd Number of RST Flags 4- Fwd Number of Push Flags 5- Bwd Number of RST Flags			Without Aggregation Features + **Number of Flows** + **Source Ports Delta**		
	Precision	Recall	F1-Score	Precision	Recall	F1-Score
Benign	98.10%± 0.05%	99.30%± 0.95%	98.69%± 0.49%	99.57%± 0.25%	99.94%± 0.04%	99.75%± 0.12%
PortScan	99.10%± 0.94%	99.39%± 0.04%	99.24%± 0.48%	99.95%± 0.03%	99.73%± 0.24%	99.84%± 0.11%
Hulk	99.94%± 0.03%	98.56%± 0.06%	99.25%± 0.03%	99.98%± 0.03%	99.50%± 0.06%	99.74%± 0.03%

Figure 7.4 shows the effect of using flow aggregation features on the recall of attack classes in a three-class classification problem. Flow aggregation shows its significance with a more complex classification problem (three-class classification).

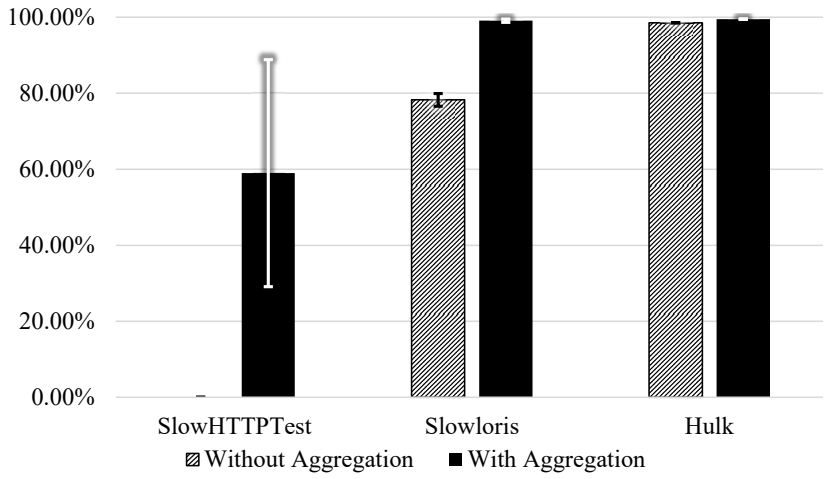


Figure 7.4
Multi-class Classification — Impact of Flow Aggregation on Recall of the Second Attack Class (Benign-PortScan-Attack)

7.4.3 Five-Class Classification Results

The final classification experiment is a five-class classification one. It combines all classes of interest into a more complex classification problem. The RFE ranking in the five-class classification is 1 for “Number of Flows” and 2 for “Source Ports Delta”, out of 30 features. This indicates the high significance of the proposed features.

The precision, recall, and F1-Score are presented in Table 7.12, for when the RFE five features are used. Two observations are noted; (i) the recall of DoS (Slowloris) rises from 1.40% to 67.81%. (ii) The recall of Dos (SlowHTTPTest) rises from 0% to 4.64% only. This is not because the new features were insignificant, but because the model classified DoS (SlowHTTPTest) as DoS (Slowloris). Flow aggregation features were used to discriminate benign-mimicking attacks from benign traffic but not to discriminate the attacks from each other. Without the aggregation features, 82.84% of DoS (SlowHTTPTest) attack instances are classified as benign, however, this drops to 58% when the flow aggregation features are used.

Table 7.12
Five-Classes Classification - 1 (5-fold cross-validation)

	Without Aggregation			With Aggregation		
RFE Selected Features	1- Fwd Mean Packet Length 2- Bwd Mean Inter-arrival time 3- Fwd mean time between the first packet and each successive packet 4- bwd mean time between the first packet and each successive packet 5- Fwd Max packet length			Without Aggregation Features + **Number of Flows** + **Source Ports Delta**		
	Precision	Recall	F1-Score	Precision	Recall	F1-Score
Benign	90.97%± 2.99%	96.77%± 0.80%	93.74%± 1.33%	95.11%± 2.11%	97.11%± 3.30%	96.05%± 1.84%
PortScan	97.12%± 0.80%	98.90%± 1.05%	98.00%± 0.38%	99.92%± 0.13%	99.41%± 0.15%	99.67%± 0.08%
Slowloris	18.89%± 37.78%	1.40% ± 2.80%	2.61% ± 5.22%	66.88%± 8.94%	67.81%± 31.08%	63.14%± 25.95%
SlowHTTP Test	0.00% ± 0.00%	0.00% ± 0.00%	0.00% ± 0.00%	34.12%± 42.81%	4.64% ± 6.51%	8.10% ± 11.21%
Hulk	93.75%± 6.47%	98.61%± 0.73%	95.98%± 3.14%	93.00%± 8.58%	99.34%± 0.15%	95.85%± 4.87%

To overcome this low recall, five more features are added by choosing the next top ones that are selected by RFE. The addition of new features results in an input layer of 10 neurons. Therefore, the ANN architecture was updated to have 8 neurons instead of 3 in the hidden layer. The results of this classification experiment are summarised in Table 7.13. The rise in the recall for the attack classes with and without flow aggregation features is as follows; from 33.94% to 80.39% and 21.45% to 64.91%, for DoS (Slowloris) and DoS (SlowHTTPTest), respectively. This behaviour is visualised in Figure 7.5. It is important to mention that while there is a rise in the recall of all classes, this rise is more significant for the attack classes that mimic benign behaviour. This demonstrates the inability of bidirectional flow features to discriminate benign-mimicking attacks solely compared to the other attack, as well as the improved effect flow aggregation features have. This is reasoned by the nature of these attacks which are crafted to bypass detection mechanisms.

Table 7.13
Five-Classes Classification - 2 (5-fold cross-validation)

	Without Aggregation			With Aggregation		
RFE Selected Features	Five RFE features + 6- Fwd Max Inter-arrival time 7- Fwd STD Inter-arrival time 8- Fwd Number of Reset Flags 9- Fwd Number of Bytes 10- Bwd Max Inter-arrival time			Without Aggregation Features + **Number of Flows** + **Source Ports Delta**		
	Precision	Recall	F1-Score	Precision	Recall	F1-Score
Benign	92.37%± 3.56%	96.34%± 0.11%	94.28%± 1.83%	97.35%± 0.53%	99.90%± 0.13%	98.61%± 0.31%
PortScan	96.48%± 0.07%	99.74%± 0.03%	98.08%± 0.04%	99.84%± 0.18%	99.59%± 0.20%	99.71%± 0.10%
Slowloris	38.91%± 47.65%	33.94%± 41.60%	36.25%± 44.41%	93.52%± 5.64%	80.39%± 2.66%	86.44%± 3.94%
SlowHTTP Test	37.52%± 45.98%	21.45%± 26.27%	27.29%± 33.44%	96.80%± 2.72%	64.91%± 16.75%	76.61%± 12.25%
Hulk	99.92%± 0.14%	98.63%± 0.67%	99.27%± 0.29%	99.88%± 0.14%	99.69%± 0.21%	99.78%± 0.15%

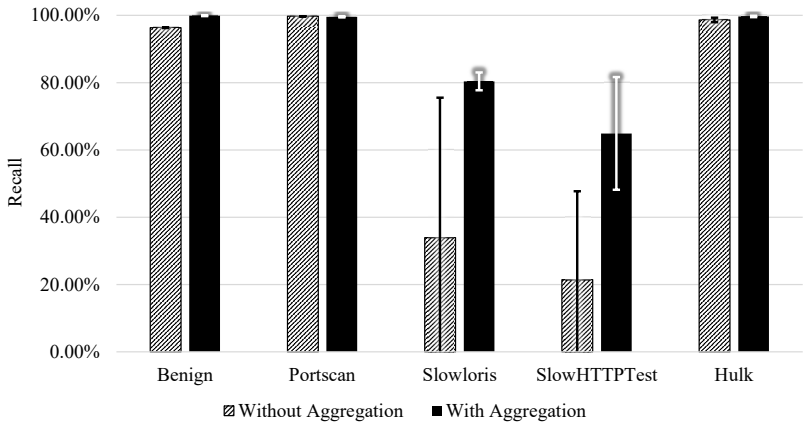


Figure 7.5
Multi-class Classification — Impact of Flow Aggregation on the Classes Recall)

7.4.4 CICIDS2017 Zero-Day Attack Detection Reassessed

In Chapter 6, the autoencoder capability to detect zero-day attacks was discussed and evaluated. Three attack categories were identified based on their detection; (i) attack classes that are detected with high accuracy, (ii) attack classes that experience a rise in the detection accuracy with lower thresholds, and (iii) attack classes that were detected with low accuracy. The last category is of interest in this section.

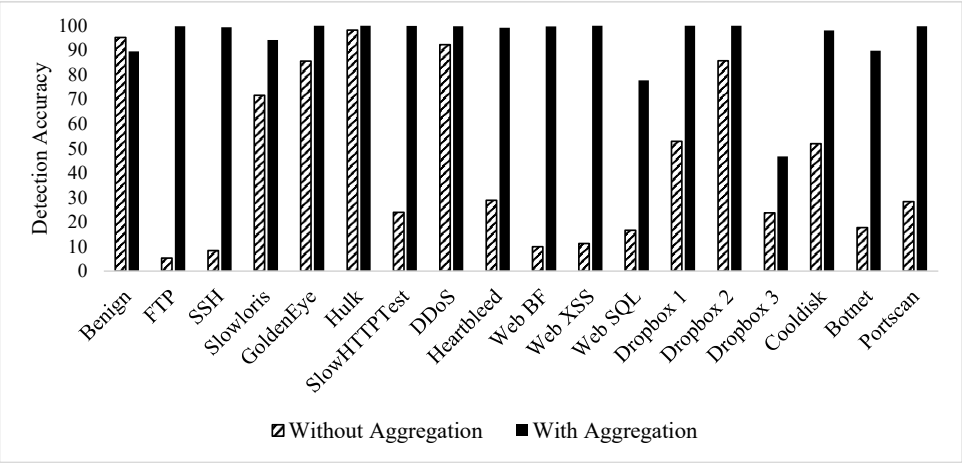
The attacks that were detected with low accuracy were hard to discriminate from benign behaviour, therefore their reconstruction error was below the zero-day threshold. After observing how flow aggregation features are effective in classifying attacks that mimic benign traffic, the autoencoder zero-day detection model is reassessed using these additional features. The same model parameters that are discussed in Chapter 6 are used.

Table 7.14 lists the zero-day detection accuracies when flow aggregation features are used alongside bidirectional flow ones. The results show a high detection rate of all the attacks, including the third category (attacks that were detected with low accuracy without flow aggregation features). Similarly, Table 7.14 shows the rise in attack detection accuracy and a decrease in benign class detection with lower thresholds, compared with Table 6.2.

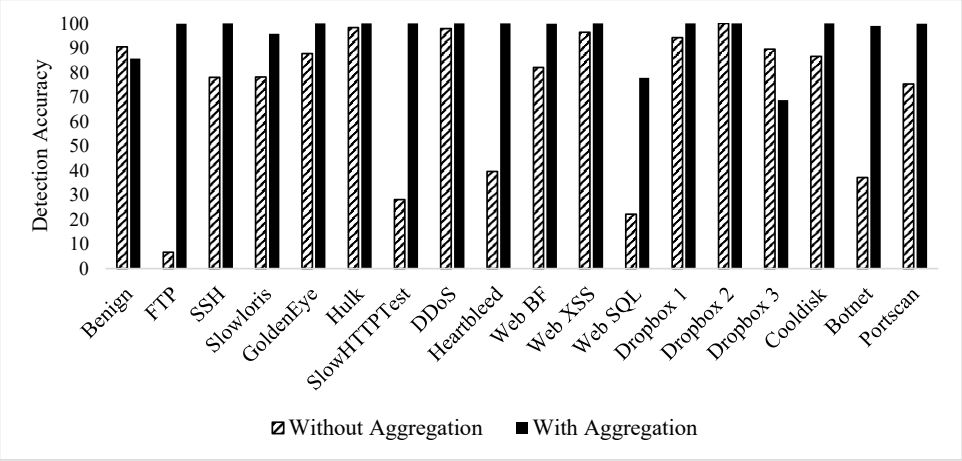
Table 7.14
Zero-Day Detection: CICIDS2017 Autoencoder Results With Flow Aggregation

Class	Accuracy		
Threshold	0.15	0.1	0.05
Benign (Validation)	89.5%	85.62%	67.59%
FTP Brute-force	99.81%	99.92%	100%
SSH Brute-force	99.37%	100%	100%
DoS (Slowloris)	94.12%	95.77%	100%
DoS (GoldenEye)	100%	100%	100%
DoS (Hulk)	100%	100%	100%
DoS (SlowHTTPTest)	99.91%	100%	100%
DDoS	99.79%	100%	100%
Heartbleed	99.13%	100%	100%
Web BF	99.7%	99.94%	100%
Web XSS	100%	100%	100%
Web SQL	77.78%	77.78%	100%
Infiltration - Dropbox 1	100%	100%	100%
Infiltration - Dropbox 2	100%	100%	100%
Infiltration - Dropbox 3	46.76%	68.68%	99.82%
Infiltration - Cooldisk	98.08%	100%	100%
Botnet	89.83%	98.98%	100%
PortScan	99.81%	99.85%	100%

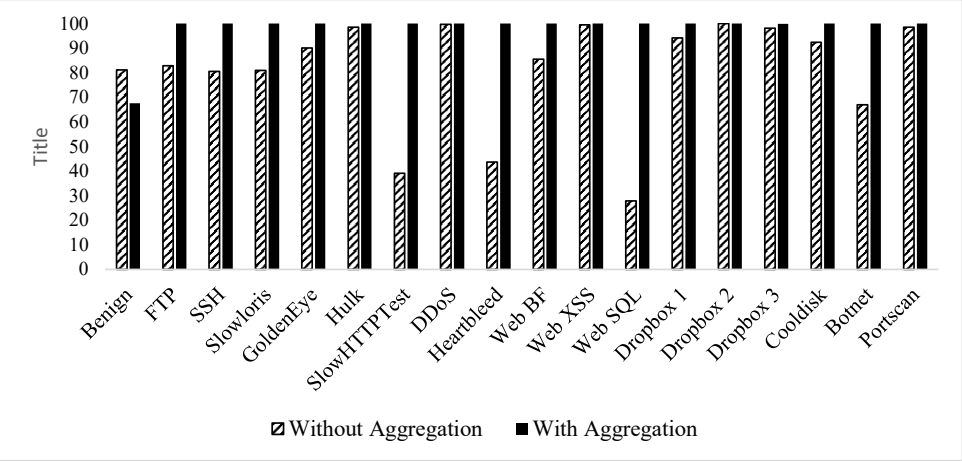
Figure 7.6 shows the effectiveness of flow aggregation features by contrasting the results that are discussed in this section versus the ones in Chapter 6. It is observed that all attacks experience a better detection accuracy when the flow aggregation features are used. This is because the additional feature abstraction provides consolidated characteristics and information. This abstraction aids in discriminating the attacks that mimic benign behaviour, therefore, the autoencoder reconstruction, in this case, is above the zero-day threshold.



(a) Threshold = 0.15



(b) Threshold = 0.1



(c) Threshold = 0.05

Figure 7.6
CICIDS2017: Zero-day Detection using Autoencoder with and without Flow Aggregation

7.5 Summary

Cyber attacks are becoming more complex and attackers use available knowledge to tailor attacks that can bypass detection methods. This chapter proposes an additional abstraction level of network flow features. The aim is to improve the cyber attack classification performance, specifically for attacks that mimic benign traffic behaviour.

Traditional network traffic features prove powerful when combined with sufficient training examples to train ML-based classifiers, and the trained models are capable of classifying cyber attacks. However, cyber attacks that are not distinctive are left undetected. The idea proposed in this chapter is based on aggregating bidirectional flows to bundles and computing bundle-specific features. Once the features are computed, the values are populated back to the bidirectional flows. The advantage of this additional feature abstraction level is that the bidirectional flows have some additional information about the flows in the same bundle (sibling flows).

In this chapter, two flow aggregation features are used and evaluated using CICIDS2017 dataset. Four cyber attack classes are used besides benign class; DoS (Slowloris), DoS (SlowHTTPTest), DoS (Hulk), and Portscan. Three classification experiments are conducted; binary classification, three-class classification, and five-class classification. The results demonstrate the validity and effectiveness of the proposed feature abstraction. This is shown in the rise of recall for attack classes. For example, the recall of the DoS (Slowloris) attack increased from 83.69% to 91.31% using binary classification, from 78.25% to 99.09% using three-class classification, and finally, from 33.94% to 80.39% using multi-class classification.

Furthermore, the flow aggregation features are used to reassess the autoencoder zero-day detection model. The results demonstrate an increase in the detection of all attacks compared to the results previously discussed in Chapter 6. The code is available on GitHub at https://github.com/AbertayMachineLearningGroup/flow-aggregation.

Chapter 8

Conclusions and Future Work

8.1 Conclusion

IDS are systems that monitor and analyse network traffic to detect anomalies and cyber attacks. Various ML techniques have been utilised in the past decade to build IDS. The predominant use of ML techniques is due to the sophistication and pace at which new cyber attacks emerge. The work presented in this thesis investigates the use of ML techniques to build special-purpose IDS. Moreover, this thesis investigates utilising novel DL techniques that are used in other research domains to build towards the next-generation IDS to improve their efficiency and effectiveness.

Based on the background presented in Chapter 2 and the analysis of the past decade of IDS in Chapter 3, the following observations are highlighted: (i) the underrepresentation of new cyber attacks in IDS datasets. (ii) the lack of special-purpose network IDS datasets, and (iii) the dominance of ML techniques usage to build IDS. Consequently, the objectives of this research are outlined.

In Chapter 4 two case studies of utilising ML techniques to build special-purpose IDS are presented; one for a SCADA network and the other for an IoT network. Six ML techniques are utilised, namely; LR, NB, k-NN, SVM, DT, and RF.

The first dataset simulates a CI that controls a water SCADA system. The dataset consists of 14 real-world scenarios that cover hardware failures, sabotage, and cyber attacks. Three experiments are conducted. The first experiment performs a binary classification (i.e., normal versus anomaly classes). The second experiment classifies the affected component (i.e., none in case of normal operation or the affected component by an anomaly or a cyber attack). Finally, the third experiment aims to classify the occurring scenario.

The evaluation of the three experiments shows that k-NN, DT, and RF outperform NB, SVM, and LR. This is owed to the non-linearity of the dataset features, which represents the CI networks setup. In this setup, normal operations and anomalies can overlap [232]. The accuracy reached 94.12% for the binary classification, 82.71% for the affected components, and 95.49% for the scenario classification. On account of the scenario overlapping problem, a confidence interval is used to report the highest probable scenario or two probable scenarios when the confidence is below the accepted interval.

The second dataset is generated and collected using a simulated IoT network that is based on MQTT protocol (a well-established machine-to-machine communication protocol) [212]. The MQTT-IoT-IDS2020 dataset covers normal operations and four cyber attack scenarios. After the dataset collection phase, three levels of feature abstraction are generated; namely, packet-based, unidirectional flow, and bidirectional flow.

The overall accuracy reached 88.55% when using packet-based features, 99.98%, and 99.9% when using unidirectional and bidirectional flow features, respectively. Furthermore, the recall demonstrated that the two classes for which performances

improved using flow-based features, are the benign class and MQTT-Brute-Force attack class. This is because in an IoT setup, generic attacks (such as scanning) are distinctive from IoT traffic behaviour. However, the challenge lies in MQTT-based attacks (protocol-based attacks) as they rely on the known MQTT communication commands, thus coinciding with normal traffic.

The main contributions of this Chapter are summarised as follows:

- Conducting three experiments on a SCADA dataset. The dataset covers real-life cyber attacks, sabotage, and hardware failure scenarios.
- Analysing the class overlapping problem when classifying anomalies in SCADA networks.
- Generating and analysing a novel dataset; MQTT-IoT-IDS2020. The dataset comprises benign, generic cyber attack and MQTT-based attack scenarios.
- Evaluating the significance of using high-level (flow-based) features to build IDS for IoT networks.
- Examining the different needs of MQTT-based versus generic attack detection, which emphasise the special setup and, thus the needs of IoT networks.

In Chapter 5 a One-Shot learning model that can learn from limited data is presented. This aims to resolve the proportional relation between the amount of required data and the size of the ML model. To this end, this chapter proposes a novel model to build IDS. This model is based on Siamese networks which differentiate between classes based on pair similarities rather than specific class discriminating features. Learning from similarities requires less data for training and provides the ability to introduce and classify new cyber attacks after training.

The Siamese network model is evaluated in three scenarios. The first one evaluates the validity of similarity-based learning for IDS usage. This is performed by assessing the ability of Siamese networks to classify attacks (i.e., differentiate between attacks) using pair similarity solely. The second scenario assesses the flexibility of the proposed

Siamese network model to classify a new cyber attacks without retraining using a few samples of this new attack. Finally, in the third scenario, the Siamese network model is evaluated to flag zero-day attacks alongside classifying known attacks (attacks that are included in training).

Four datasets are used for evaluation; namely, SCADA, CICIDS2017, NSL-KDD, and KDD Cup'99. In the classification scenario. the accuracy of the first dataset (SCADA) is 76.06%. However, it is seen that the classes either have high classification accuracy (reaching 100%) or a low accuracy (less than 50%). This is due to the class overlapping problem of the SCADA dataset and its multi-label nature, where classes are not mutually exclusive. The Siamese network cannot discriminate highly overlapping classes only using similarity. By observing the classification results, it is proved that Siamese network can discriminate classes that overlap with at most 7 other classes (in a dataset of 14 classes).

For the CICIDS2017, the overall classification accuracy is 83.74%. The different attack classes detection accuracies are 96.08%, 75.17%, 80.05%, and 76.55%, respectively. The results show that using only one pair to classify attacks is not enough. This is due to the pair selection randomness, which increases the probability of selecting a representable pair as the number of pairs increases. Therefore, multiple instances are used and majority voting technique decides on the class label. Similar behaviour was noted for the NSL-KDD and KDD Cup'99 datasets with an overall accuracy of 91.01% and 87.99%, respectively.

In the second scenario, an attack class is excluded from the Siamese network model training. During the evaluation, a few instances from the excluded class are used as labelled instances of a new attack. The excluded attack mimics the situation when a new cyber attack is detected, but there are no enough instances to retrain IDS to detect it. In this case, the proposed One-Shot learning model aims to classify this new attack, using pair similarities, in the interim time between identifying this new attack and collecting enough instances to retrain an IDS.

For the CICIDS2017 dataset, the overall accuracy is 81.28% and 82.5%, when excluding the SSH and FTP attack classes, respectively. The overall accuracy demonstrates that the network performance is not disturbed by the attack class addition post training when compared to 83.74% when all classes are used during training. Moreover, the new classes detection accuracies are 73.03% and 70.03% for SSH and FTP, respectively.

The evaluation of the NSL-KDD demonstrated that it outperforms that of the KDD Cup'99. The detection accuracies of the DoS attack (when excluded from training) are 40.28% for the KDD Cup'99 and 78.87% for the NSL-KDD. This is because the NSL-KDD dataset is an enhanced version of the KDD Cup'99. Given that the new class is not included in training, having a better representation of instances shows a better performance, therefore, NSL-KDD performance outperforms KDD Cup'99.

This demonstrates an observation about the Siamese network training. Since the training and evaluation are based on pairing instances from different classes, the more representative the instances are, the better the Siamese network performance.

Finally, for the zero-day detection scenario, the Siamese network was capable of discriminating 84.8% of the SSH and 94.17% of the DoS (Hulk) attacks in CICIDS2017. The FTP class was detected as zero-day attack with low rate, however, only 4.83% was classified as normal. This indicates that the Siamese network classified the new attack as another attack but not benign behaviour. For the NSL-KDD and KDD Cup'99 datasets 85.85%, and 72.83% of the R2L class are correctly flagged as unknown, respectively. Similar behaviour to the CICIDS2017 dataset classes was observed for the other attack classes in the NSL-KDD and KDD Cup'99 datasets. This indicates the restricted capability of detecting zero-day attacks for classes that are highly distinctive from other attack classes.

The main contributions of this Chapter can be summarised as follows:

- Proposing a novel Siamese network model to classify attacks based on learning from similarity (few samples-based standard classifier).

- Implementing a One-Shot Siamese network and evaluating its performance to detect a new cyber attack class based on a few labelled samples without retraining.
- Evaluating the use of Siamese network to detect Zero-Day attacks.
- Demonstrating the need for distinctive samples to boost the Siamese network performance.

Chapter 6 focuses on building a model that is capable of detecting zero-day attacks. The proposed model leverages the encoding-decoding capabilities of autoencoders. Benign traffic is used to train the model and relying on the reconstruction error the zero-day attacks are detected. Furthermore, the proposed autoencoder performance is compared with a One-Class SVM.

The CICIDS2017 zero-day detection accuracy reaches 90.01%, 98.43%, 98.47%, and 99.67% for DoS (GoldenEye), DoS (Hulk), PortScan and DDoS attacks, respectively. The KDD Cup'99 dataset detection accuracy reached 95.21% and NSL-KDD dataset detection accuracy reaches 92.96%.

The one-class SVM model shows its high performance in detecting zero-day attacks for KDD Cup'99 and NSL-KDD datasets and the distinctive attack classes from the CICIDS2017 dataset. Compared to One-Class SVM, which has proven its effectiveness and high accuracy in novelty detection in the literature, the autoencoder demonstrates its better detection accuracy. Both models demonstrate low FPR. Finally, the CICIDS2017 classes that mimic benign traffic behaviour (DoS (Slowloris), DoS (SlowHTTPTest)) experience lower detection rates by both the autoencoder and the One-Class SVM models. This is because these attacks are launched with a behaviour that is similar to benign traffic, thus, their reconstruction error is low.

The main contributions of this chapter are summarised as follows:

- Proposing and implementing an autoencoder model for zero-day detection IDS.
- Building an outlier detection One-Class SVM model.

- Comparing the performance of the One-Class SVM model as a baseline outlier-based detector to the proposed autoencoder model.

Chapter 7 proposes an additional level of feature abstraction, named "Flow Aggregation", to assist in detecting benign-mimicking attacks. This additional level of feature abstraction benefits from the cumulative information of the flow communication between nodes.

In this chapter, the focus is on the attacks that are hard to detect using bidirectional flow features. Three classification experiments are conducted; binary classification, three-class classification, and five-class classification. The results demonstrate the validity and effectiveness of the proposed feature abstraction. This is shown in the rise of recall for attack classes. For example, the recall of the DoS (Slowloris) attack increased from 83.69% to 91.31% using binary classification, from 78.25% to 99.09% using three-class classification, and finally, from 21.45% to 99.69% using multi-class classification.

Finally, the flow aggregation features are used to reassess the autoencoder zero-day detection model. The results demonstrate an increase in the detection of all attacks, including benign mimicking ones, compared to the results presented in Chapter 6.

The main contribution of this chapter is summarised as follows:

- Introducing a higher level of abstraction for network traffic analysis by proposing novel features to describe bundles of flows.
- Assessing the performance improvements in binary classification of cyber attacks when these new features are utilised, particularly for attacks that mimic benign network traffic.
- Assessing the performance improvements in multi-class classification of cyber attacks when these new features are utilised.

Within this thesis, different models are proposed and evaluated to address the research questions as follows:

RQ1: How can Machine Learning be utilised to detect anomalies and attacks in special-purpose networks (IoT and CI)?

For special-purpose IDS, ML can be utilised to detect anomalies and cyber attacks as demonstrated in Chapter 4. Based on the chapter findings, special-purpose networks have unique architecture based on their application. As a result, special-purpose IDS development face different challenges compared with general-purpose ones. Probabilistic models and confidence intervals can be used to overcome the overlapping of anomaly scenarios. Also, flow-based features are better suited to discriminate protocol-specific attacks.

RQ2: In an attempt to reduce the burden of needing to generate/collect large volumes of data, can IDS models train using limited-size datasets?

Based on the Siamese network model proposed and evaluated in Chapter 5, IDS can be trained using limited data, based on similar and dissimilar pairs. A One-Shot learning paradigm using Siamese networks proved its applicability and effectiveness to develop IDS. The experiments results demonstrate that Siamese networks perform better when datasets contain representable instances from each class and minimise class overlap.

RQ3: In order to reduce the interim period between identifying a new cyber attack and detecting it, is there potential to build IDS that can detect new cyber attacks without retraining?

The similarity-based learning using Siamese network presented in Chapter 5 was found to benefit IDS development process by enabling the detection of new attacks after initial training. Detection of new attacks is possible when a few instances are available. This serves as an appropriate interim detection mechanism until more samples are available and retraining takes place.

RQ4: How can non-conventional DL techniques provide improved robustness and accuracy for IDS when detecting zero-day attacks?

In Chapter 6, autoencoders show their ability to serve as zero-day detection models by training on normal traffic instances only and relying on their encoding-decoding capabilities to flag attacks. Furthermore, in Chapter 7 flow aggregation features are proposed. These features provide an additional level of feature abstraction that improves the classification performance for complex attacks. This is demonstrated in classifying benign-mimicking attacks and detecting zero-day attacks that are hard to flag.

8.2 Future Work

The research presented in this thesis can be extended as follows.

8.2.1 Special-Purpose Network IDS

The lack of special-purpose network datasets, and special-purpose IDS accordingly, are outlined in this thesis. To fill this gap, different ML techniques are used to build IDS for a SCADA and IoT networks.

Building on the work discussed in this thesis, investigation of other IoT-based attacks is needed. This investigation will help in examining the unique requirements of special-purpose IDS. This process involves the generation of new special-purpose datasets that comprise these attacks alongside generic ones. Firstly, the generation setup and platform of MQTT-IoT-IDS2020 dataset can be extended to include additional components and scenarios. This will result in a larger corpus of data and a quick generation of datasets.

Secondly, utilising multi-label classification [342], where an instance can belong to one or more classes, to build special-purpose IDS. This would assist in improving the

IDS performance and overcome the co-existable, inclusive and overlapping nature of anomaly classes in these networks.

Thirdly, based on the constantly changing and different requirements for CI and IoT networks, exploring the opportunities of transfer learning to assist in accelerating the special-purpose IDS development is needed. This research will involve both methods of standardising dataset generation, processing, and the applicability of different transfer learning approaches.

8.2.2 Few-Shot Learning

This thesis proposed the use of Siamese networks as a One-Shot/Few-Shot learning model to build a novel IDS. To the best of the author's knowledge, this is the first attempt to leverage these learning paradigms for IDS purposes. The Siamese network demonstrated its ability to learn cyber attack similarities. As a result, the Siamese network model is capable of detecting new attacks using a few samples and the ability to flag unknown zero-day attacks.

Firstly, "Triplet Networks", which are based on Siamese networks, have been proposed in other domains to learn similar and dissimilar relation concurrently [343]. A Triplet network is composed of three identical networks that train simultaneously. Unlike Siamese networks that learn from similar and dissimilar pairs, Triplet Networks learn from a set of three instances (two of the same class and one from a different class). These three instances are called; positive, anchor, negative. During training, triplet loss is used to reduce the distance between the two instances from the same class while increasing that of the instance from the different class [344]. Investigating the use of Triplet networks for IDS can improve the detection accuracy in the interim time between identifying a new attack and detecting it.

Secondly, the pair generation is still an open research issue [288]. Research in this direction will involve studying different approaches to pair selection, other than random selection, and their effect on the Siamese network model.

Thirdly, the examination of Siamese network adaptability can be extended. This can be done by evaluating the performance of introducing multiple attacks post training. The experiments presented in this thesis outlined the Siamese network ability to classify a new attack without retraining. Examining the model capability to multiple attacks will extend its usage and gain.

8.2.3 Zero-Day Attack Detection

In this thesis, autoencoders are used to detect zero-day attacks by relying on their encoding-decoding capabilities. The autoencoder performance model is compared with the novelty detection model; One-Class SVM. This experiment can be extended to include other IDS datasets. Furthermore, other non-conventional ML techniques can be considered to detect zero-day attacks. For example, the use of LSTM memorisation capabilities to learn normal operation patterns can be examined.

8.2.4 Flow Aggregation

Flow aggregation is a new direction for extracting high-level network flow features. In this thesis, two new features are proposed and their significance is evaluated. Introducing other flow aggregation-based features can assist in the detection of benign-mimicking attacks. These features represent the characteristics of the communication between different nodes during normal and attack operations.

Flow aggregation features can be proposed and evaluated on other communication networks. An investigation of their impact on ad-hoc networks, industrial protocols, etc. allows benefiting from the concept of the additional level of feature abstraction.

Further work can include the generation of the flow aggregation features in a sliding time-window scheme. Different time windows can be applied to grouping/bundling the network flows. This experiment should examine both the impact of windowing on the IDS detection accuracy and the trade-off between different window sizes and the detection of various benign-mimicking attacks.

Finally, the potential of evaluating the models proposed in this thesis in real testing setup would provide additional insights. This would further highlight the needs and requirements in different operational scenarios, as opposed to benchmark datasets evaluation.

References

[1] C. E. Bondoc and T. G. Malawit, "Cybersecurity for higher education institutions: Adopting regulatory framework." *Global Journal of Engineering and Technology Advances*, vol. 2, no. 3, p. 16, 2020. https://doi.org/10.30574/gjeta.2020.2.3.0013

[2] I. Nai-Fovino, R. Neisse, A. Lazari, G.-L. Ruzzante, N. Polemi, and M. Figwer, "European cybersecurity centres of expertise map - definitions and taxonomy," *Luxembourg, Publications Office of the European Union*, 2018. https://doi.org/10.2760/622400

[3] I. Butun, S. D. Morgera, and R. Sankar, "A survey of intrusion detection systems in wireless sensor networks." *IEEE Communications Surveys & Tutorials*, vol. 16, no. 1, pp. 266–282, 2014. https://doi.org/10.1109/SURV.2013.050113.00191

[4] D. S. Berman, A. L. Buczak, J. S. Chavis, and C. L. Corbett, "A survey of deep learning methods for cyber security." *Information*, vol. 10, no. 4, p. 122, 2019. https://doi.org/10.3390/info10040122

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

[6] 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. 105124, 2020. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0950705119304897. https://doi.org/10.1016/j.knosys.2019.105124

[7] Cisco, "Cisco annual internet report - Cisco annual internet report (2018–2023) white paper," 9 March 2020, Accessed: 9 December, 2020. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html

[8] U. P. Ramakrishnan and J. K. Tandon, "The evolving landscape of cyber threats." *Vidwat: The Indian Journal of Management*, vol. 11, 2018.

[9] P. Kreimel, O. Eigner, and P. Tavolato, "Anomaly-based detection and classification of attacks in cyber-physical systems," in *Proceedings of the 12th International Conference on Availability, Reliability and Security*. ACM, 2017, pp. 1–6. https://doi.org/10.1145/3098954.3103155

[10] S. Hameed, F. I. Khan, and B. Hameed, "Understanding security requirements and challenges in internet of things (IoT): A review." *Journal of Computer Networks and Communications*, vol. 2019, pp. 1–14, 2019. https://doi.org/10.1155/2019/9629381

[11] B. Li, J. Springer, G. Bebis, and M. H. Gunes, "A survey of network flow applications." *Journal of Network and Computer Applications*, vol. 36, no. 2, pp. 567–581, 2013. https://doi.org/10.1016/j.jnca.2012.12.020

[12] Y. Ayrour, A. Raji, and M. Nassar, "Modelling cyber-attacks: A survey study." *Network Security*, vol. 2018, no. 3, pp. 13–19, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1353485818300254. https://doi.org/10.1016/S1353-4858(18)30025-4

[13] H. Hindy, D. Brosset, E. Bayne, A. Seeam, C. Tachtatzis, R. Atkinson, and X. Bellekens, "A taxonomy of network threats and the effect of current datasets on intrusion detection systems." *IEEE Access*, vol. 8, pp. 104 650–104 675, 2020. https://doi.org/10.1109/ACCESS.2020.3000179

[14] H. Hindy, E. Hodo, E. Bayne, A. Seeam, R. Atkinson, and X. Bellekens, "A taxonomy of malicious traffic for intrusion detection systems." in *2018 International Conference On Cyber Situational Awareness, Data Analytics*

And Assessment (Cyber SA). IEEE, 2018, pp. 1–4. https://doi.org/10.1109/CyberSA.2018.8551386

[15] H. Hindy, D. Brosset, E. Bayne, A. Seeam, and X. Bellekens, "Improving SIEM for critical SCADA water infrastructures using machine learning." in *SECPRE 2018, CyberICPS 2018*, S. K. Katsikas, F. Cuppens, N. Cuppens, C. Lambrinoudakis, A. Antón, S. Gritzalis, J. Mylopoulos, and C. Kalloniatis, Eds. Cham: Springer International Publishing, 2019, pp. 3–19. https://doi.org/10.1007/978-3-030-12786-2_1

[16] H. Hindy, C. Tachtatzis, R. Atkinson, E. Bayne, and X. Bellekens, "MQTT-IoT-IDS2020: MQTT internet of things intrusion detection dataset." 2020. [Online]. Available: https://ieee-dataport.org/open-access/mqtt-iot-ids2020-mqtt-internet-things-intrusion-detection-dataset. https://doi.org/10.21227/bhxy-ep04

[17] H. Hindy, E. Bayne, M. Bures, R. Atkinson, C. Tachtatzis, and X. Bellekens, "Machine learning based IoT intrusion detection system: an MQTT case study (MQTT-IoT-IDS2020 dataset)." in *12th International Network Conference (INC 2020)*. Springer, 2020.

[18] H. Hindy, C. Tachtatzis, R. Atkinson, E. Bayne, and X. Bellekens, "Developing a siamese network for intrusion detection systems," in *Proceedings of the 1st Workshop on Machine Learning and Systems*, ser. EuroMLSys '21. New York, NY, USA: ACM, 2021, p. 120–126. https://doi.org/10.1145/3437984.3458842

[19] H. Hindy, C. Tachtatzis, R. Atkinson, D. Brosset, M. Bures, I. Andonovic, C. Michie, and X. Bellekens, "Leveraging Siamese networks for One-Shot intrusion detection model." *arXiv preprint arXiv:2006.15343*, 2020.

[20] H. Hindy, R. Atkinson, C. Tachtatzis, J.-N. Colin, E. Bayne, and X. Bellekens, "Utilising deep learning techniques for effective zero-day attack detection." *Electronics*, vol. 9, no. 10, p. 1684, 2020. https://doi.org/10.3390/electronics9101684

[21] H. Hindy, R. Atkinson, C. Tachtatzis, E. Bayne, M. Bures, and X. Bellekens, "Utilising flow aggregation to classify benign imitating attacks." *Sensors*, vol. 21, no. 5, p. 1761, 2021. https://doi.org/10.3390/s21051761

[22] A. Fernández, S. García, M. Galar, R. C. Prati, B. Krawczyk, and F. Herrera, *Learning from imbalanced data sets*. Springer, 2018, vol. 10.

[23] M. A. Ferrag, L. Maglaras, S. Moschoyiannis, and H. Janicke, "Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study." *Journal of Information Security and Applications*, vol. 50, p. 102419, 2020. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2214212619305046. https://doi.org/10.1016/j.jisa.2019.102419

[24] CAIDA, "Flow types," November 2019, Accessed: 25 June, 2020. [Online]. Available: https://www.caida.org/research/traffic-analysis/flowtypes/index.xml

[25] C. Ehret and U. Ultes-Nitsche, "Immune system based intrusion detection system." in *Innovative Minds (Information Systems Security Association-ISSA 2008), Johannesburg, South Africa, July 2008*, 2008.

[26] B. Atli, "Anomaly-based intrusion detection by modeling probability distributions of flow characteristics." Master of Science in Technology, School of Electrical Engineering, 23 October 2017. [Online]. Available: http://urn.fi/URN:NBN:fi:aalto-201710307348

[27] X. J. Bellekens, C. Tachtatzis, R. C. Atkinson, C. Renfrew, and T. Kirkham, "GLoP: Enabling massively parallel incident response through GPU log processing." in *Proceedings of the 7th International Conference on Security of Information and Networks*, 2014, pp. 295–301. https://doi.org/10.1145/2659651.2659700

[28] W. Wang, Y. Sheng, J. Wang, X. Zeng, X. Ye, Y. Huang, and M. Zhu, "HAST-IDS: Learning hierarchical spatial-temporal features using deep neural networks to improve intrusion detection." *IEEE Access*, vol. 6, pp. 1792–1806, 2018. https://doi.org/10.1109/ACCESS.2017.2780250

[29] DNSStuff, "IDS vs. IPS: What's the difference?" June 28, 2019, Accessed: 21 October, 2020. [Online]. Available: https://www.dnsstuff.com/ids-vs-ips

[30] A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, "Survey of intrusion detection systems: Techniques, datasets and challenges." *Cybersecurity*, vol. 2, no. 1, p. 20, 2019. https://doi.org/10.1186/s42400-019-0038-7

[31] T. T. T. Nguyen and G. Armitage, "A survey of techniques for internet traffic classification using machine learning." *IEEE Communications Surveys & Tutorials*, vol. 10, no. 4, pp. 56–76, 2008. https://doi.org/10.1109/SURV.2008.080406

[32] A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection." *IEEE Communications surveys & tutorials*, vol. 18, no. 2, pp. 1153–1176, 2016. https://doi.org/10.1109/COMST.2015.2494502

[33] T. Hamed, J. B. Ernst, and S. C. Kremer, *A Survey and Taxonomy of Classifiers of Intrusion Detection Systems.*, ser. Computer and Network Security Essentials. Cham: Springer International Publishing, 2018, pp. 21–39. https://doi.org/10.1007/978-3-319-58424-9_2

[34] G. Sunil, "Logging and monitoring to detect network intrusions and compliance violations in the environment." *SANS Institute InfoSec Reading Room. SANS Institute*, pp. 1–42, 2012.

[35] F. Blog, "Categorical vs numerical data: 15 key differences & similarities," 2020, Accessed: 12 January, 2021. [Online]. Available: https//www.formpl.us/blog/categorical-numerical-data

[36] S. Kumar, "7 ways to handle missing values in machine learning," 24 July 2020, Accessed: 23 October, 2020. [Online]. Available: https://towardsdatascience.com/7-ways-to-handle-missing-values-in-machine-learning-1a6326adf79e

[37] G. Rosati, "Dealing with missing data," 28 July 2018, Accessed: 23

October, 2020. [Online]. Available: https://towardsdatascience.com/dealing-with-missing-data-17f8b5827664

[38] V. Sharma, "Handling categorical data in machine learning models," 20 February 2019, Accessed: 23 October, 2020. [Online]. Available: https://www.pluralsight.com/guides/handling-categorical-data-in-machine-learning-models

[39] A. Bhandari, "Feature scaling — standardization vs normalization," 3 April 2020, Accessed: 23 October, 2020. [Online]. Available: https://www.analyticsvidhya.com/blog/2020/04/feature-scaling-machine-learning-normalization-standardization/

[40] T. Stöttner, "Why data should be normalized before training a neural network," 16 May 2019, Accessed: 12 January, 2021. [Online]. Available: https://towardsdatascience.com/why-data-should-be-normalized-before-training-a-neural-network-c626b7f66c7d

[41] M. E. Aminanto, R. Choi, H. C. Tanuwidjaja, P. D. Yoo, and K. Kim, "Deep abstraction and weighted feature selection for Wi-Fi impersonation detection." *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 3, pp. 621–636, 2018. https://doi.org/10.1109/TIFS.2017.2762828

[42] S. M. Ghaffarian and H. R. Shahriari, "Software vulnerability analysis and discovery using machine-learning and data-mining techniques: A survey." *ACM Computing Surveys (CSUR)*, vol. 50, no. 4, p. 56, 2017. https://doi.org/10.1145/3092566

[43] H. Liu and B. Lang, "Machine learning and deep learning methods for intrusion detection systems: A survey." *Applied Sciences*, vol. 9, no. 20, p. 4396, 2019.

[44] T. Shah, "About train, validation and test sets in machine learning," 6 December 2017, Accessed: Jul 14, 2021. [Online]. Available: https://towardsdatascience.com/train-validation-and-test-sets-72cb40cba9e7

[45] G. Tennenholtz, T. Zahavy, and S. Mannor, "Train on validation: squeezing the data lemon," *arXiv preprint arXiv:1802.05846*, 2018.

[46] J. Brownlee, "What is the difference between test and validation datasets?" 13 July 2017, Accessed: Jul 12, 2021, 2021. [Online]. Available: https://machinelearningmastery.com/difference-test-validation-datasets/

[47] A. Bronshtein, "Train/test split and cross validation in Python," 17 May 2017, Accessed: 27 January, 2021. [Online]. Available: https://towardsdatascience.com/train-test-split-and-cross-validation-in-python-80b61beca4b6

[48] T. Hastie, R. Tibshirani, and J. Friedman, *Model Assessment and Selection: Cross-Validation.*, 2nd ed., ser. The Elements of Statistical Learning: Data Mining, Inference, and Prediction. New York: Springer Series in Statistics, February 2009, pp. 219–249.

[49] P. Cunningham and S. J. Delany, "Algorithmic bias and regularisation in machine learning." *arXiv preprint arXiv:2005.09052*, 2020.

[50] R. Khandelwal, "L1 and L2 regularization," 4 November 2018, Accessed: 10 March, 2021. [Online]. Available: https://medium.datadriveninvestor.com/l1-l2-regularization-7f1b4fe948f2

[51] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting." *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[52] Y. Chen, Y. Li, X.-Q. Cheng, and L. Guo, "Survey and taxonomy of feature selection algorithms in intrusion detection system." in *International Conference on Information Security and Cryptology*. Springer, 2006, pp. 153–167. https://doi.org/10.1007/11937807_13

[53] S. Rezaei and X. Liu, "Deep learning for encrypted traffic classification: An overview." *IEEE Communications Magazine*, vol. 57, no. 5, pp. 76–81, 2019. https://doi.org/10.1109/MCOM.2019.1800819

[54] J. M. Cadenas, M. C. Garrido, and R. Martínez, "Feature subset selection filter–wrapper based on low quality data." *Expert Systems with Applications*, vol. 40, no. 16, pp. 6241–6252, 2013. [Online]. Available:

http://www.sciencedirect.com/science/article/pii/S0957417413003497. https://doi.org/10.1016/j.eswa.2013.05.051

[55] T. M. Phuong, Z. Lin, and R. B. Altman, "Choosing SNPs using feature selection." in *2005 IEEE Computational Systems Bioinformatics Conference (CSB'05)*. IEEE, 2005, pp. 301–309. https://doi.org/10.1109/CSB.2005.22

[56] J. C. H. Hernandez, B. Duval, and J.-K. Hao, "A genetic embedded approach for gene selection and classification of microarray data." in *European Conference on Evolutionary Computation, Machine Learning and Data Mining in Bioinformatics*. Springer, 2007, pp. 90–101. https://doi.org/10.1007/978-3-540-71783-6_9

[57] S. Axelsson, "The base-rate fallacy and the difficulty of intrusion detection." *ACM Transactions on Information and System Security (TISSEC)*, vol. 3, no. 3, pp. 186–205, 2000. https://doi.org/10.1145/357830.357849

[58] E. Vasilomanolakis, S. Karuppayah, M. Mühlhäuser, and M. Fischer, "Taxonomy and survey of collaborative intrusion detection," *ACM Computing Surveys (CSUR)*, vol. 47, no. 4, May 2015. https://doi.org/10.1145/2716260

[59] E. Hodo, X. Bellekens, A. Hamilton, C. Tachtatzis, and R. Atkinson, "Shallow and deep networks intrusion detection system: A taxonomy and survey." *arXiv preprint arXiv:1701.02145*, pp. 1–43, 2017.

[60] A. P. Bradley, "The use of the area under the ROC curve in the evaluation of machine learning algorithms." *Pattern Recognition*, vol. 30, no. 7, pp. 1145–1159, 1997. https://doi.org/10.1016/S0031-3203(96)00142-2

[61] S. Narkhede, "Understanding AUC - ROC curve," 2018, Accessed: 29 July, 2019. [Online]. Available: https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5

[62] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization." in *ICISSP*, 2018, pp. 108–116. https://doi.org/10.5220/0006639801080116

[63] Canadian Institute for Cybersecurity, "Intrusion detection evaluation dataset (CICIDS2017)," Accessed: 15 June, 2018. [Online]. Available: http://www.unb.ca/cic/datasets/ids-2017.html

[64] Canadian Institute for Cybersecurity, "CIC DoS dataset," Accessed: 15 June, 2018. [Online]. Available: http://www.unb.ca/cic/datasets/dos-dataset.html

[65] G. Creech and J. Hu, "ADFA IDS dataset," 2017, Accessed: 13 May, 2019. [Online]. Available: http://www.azsecure-data.org/

[66] G. Creech and J. Hu, "Generation of a new IDS test dataset: Time to retire the KDD collection." in *2013 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, 2013, pp. 4487–4492. https://doi.org/10.1109/WCNC.2013.6555301

[67] M. J. Turcotte, A. D. Kent, and C. Hash, "Unified host and network data set." *Data Science For Cyber-security*, vol. 3, pp. 1–16, 2018. https://doi.org/10.1142/9781786345646_001

[68] S. Behal and K. Kumar, "Measuring the impact of DDoS attacks on web services-a realtime experimentation." *International Journal of Computer Science and Information Security*, vol. 14, no. 9, p. 323, 2016.

[69] J. J. Santanna, R. van Rijswijk-Deij, R. Hofstede, A. Sperotto, M. Wierbosch, L. Z. Granville, and A. Pras, "Booters — an analysis of DDoS-as-a-service attacks." in *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*. IEEE, 2015, pp. 243–251. https://doi.org/10.1109/INM.2015.7140298

[70] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, "Towards generating real-life datasets for network intrusion detection." *IJ Network Security*, vol. 17, no. 6, pp. 683–701, 2015.

[71] Canadian Institute for Cybersecurity, "Botnet dataset," Accessed: 15 June, 2018. [Online]. Available: http://www.unb.ca/cic/datasets/botnet.html

[72] A. A. Tobi, "STA2018," September, 2018, Accessed: 10 October, 2018. [Online]. Available: https://github.com/elud074/STA2018

[73] S. García, M. Grill, J. Stiborek, and A. Zunino, "An empirical comparison of botnet detection methods." *Computers & Security*, vol. 45, pp. 100–123, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167404814000923. https://doi.org/10.1016/j.cose.2014.05.011

[74] Canadian Institute for Cybersecurity, "Intrusion detection evaluation dataset (ISCXIDS2012)," Accessed: 15 June, 2018. [Online]. Available: http://www.unb.ca/cic/datasets/ids.html

[75] RIPE Network Coordination Center, "The Waikoto internet trace storage project dataset," 2009, Accessed: 5 June, 2020. [Online]. Available: https://labs.ripe.net/datarepository/data-sets/the-waikato-internet-traffic-storage-wits-passive-datasets

[76] Center for Applied Internet Data Analysis, "The CAIDA UCSD "DDoS attack 2007" dataset," 2007, Accessed: 05 May, 2020. [Online]. Available: https://www.caida.org/data/passive/ddos-20070804_dataset.xml

[77] Canadian Institute for Cybersecurity, "NSL-KDD dataset," Accessed: 15 June, 2018. [Online]. Available: http://www.unb.ca/cic/datasets/nsl.html

[78] S. Hettich and S. D. Bay, "The UCI KDD archive," 1999, Accessed: 15 June, 2018. [Online]. Available: http://kdd.ics.uci.edu

[79] L. Laboratory, "MIT lincoln laboratory: DARPA intrusion detection evaluation," Accessed: 15 June, 2018. [Online]. Available: https://www.ll.mit.edu/ideval/data

[80] N. Koroniotis, N. Moustafa, E. Sitnikova, and B. Turnbull, "Towards the development of realistic botnet dataset in the Internet of Things for network forensic analytics: Bot-IoT dataset." *Future Generation Computer Systems*, vol. 100, pp. 779–796, 2019. [Online]. Available: http://www.sciencedirect.com/

science/article/pii/S0167739X18327687. https://doi.org/10.1016/j.future.2019.05.041

[81] P. M. Laso, D. Brosset, and J. Puentes, "Dataset of anomalies and malicious acts in a cyber-physical subsystem." *Data in Brief*, vol. 14, pp. 186–191, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2352340917303402. https://doi.org/10.1016/j.dib.2017.07.038

[82] M. Miettinen, S. Marchal, I. Hafeez, N. Asokan, A.-R. Sadeghi, and S. Tarkoma, "IoT sentinel: Automated device-type identification for security enforcement in IoT." in *IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2017, pp. 2177–2184. https://doi.org/10.1109/ICDCS.2017.283

[83] Canadian Institute for Cybersecurity, "Tor-nonTor dataset," Accessed: 15 June, 2018. [Online]. Available: http://www.unb.ca/cic/datasets/tor.html

[84] Canadian Institute for Cybersecurity, "VPN-nonVPN dataset," Accessed: 15 June, 2018. [Online]. Available: http://www.unb.ca/cic/datasets/vpn.html

[85] NETRESEC, "SCADA / ICS PCAP files from 4SICS," Accessed: 13 May, 2019. [Online]. Available: https://www.netresec.com/?page=PCAP4SICS

[86] I. Sharafaldin, A. Gharib, A. H. Lashkari, and A. A. Ghorbani, "Towards a reliable intrusion detection benchmark dataset." *Software Networking*, vol. 2018, no. 1, pp. 177–200, 2018. https://doi.org/10.13052/jsn2445-9739.2017.009

[87] T. S. Group, "DEFCON 8, 10 and 11," 2000, Accessed: 15 June, 2018. [Online]. Available: http://cctf.shmoo.com/

[88] Center for Applied Internet Data Analysis, "CAIDA data," Accessed: 15 June, 2018. [Online]. Available: http://www.caida.org/data/index.xml

[89] L. Lawrence Berkeley National Laboratory and I. International Computer Science Institute, "LBNL/ICSI enterprise tracing project," 2005, Accessed:

15 June, 2018. [Online]. Available: http://www.icir.org/enterprise-tracing/Overview.html

[90] B. Sangster, T. J. O'Connor, T. Cook, R. Fanelli, E. Dean, C. Morrell, and G. J. Conti, "Toward instrumenting network warfare competitions to generate labeled datasets." in *CSET*. Berkeley, CA: Usenix, The Advanced Computing System Association), 2009.

[91] J. Song, H. Takakura, Y. Okabe, M. Eto, D. Inoue, and K. Nakao, "Statistical analysis of honeypot data and building of Kyoto 2006 dataset for NIDS evaluation." in *Proceedings of the First Workshop on Building Analysis Datasets and Gathering Experience Returns for Security*. ACM, 2011, pp. 29–36. https://doi.org/10.1145/1978672.1978676

[92] A. Sperotto, R. Sadre, F. V. Vliet, and A. Pras, "A labeled data set for flow-based intrusion detection." in *International Workshop on IP Operations and Management*. Springer, 2009, pp. 39–50. https://doi.org/10.1007/978-3-642-04968-2_4

[93] S. Prusty, B. N. Levine, and M. Liberatore, "Forensic investigation of the OneSwarm anonymous filesharing system." in *Proceedings of the 18th ACM conference on Computer and communications security*. ACM, 2011, pp. 201–214. https://doi.org/10.1145/2046707.2046731

[94] M. Ring, S. Wunderlich, D. Scheuring, D. Landes, and A. Hotho, "A survey of network-based intrusion detection data sets." *Computers & Security*, vol. 86, pp. 147–167, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S016740481930118X. https://doi.org/10.1016/j.cose.2019.06.005

[95] E. K. Viegas, A. O. Santin, and L. S. Oliveira, "Toward a reliable anomaly-based intrusion detection in real-world environments." *Computer Networks*, vol. 127, pp. 200–216, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128617303225. https://doi.org/10.1016/j.comnet.2017.08.013

[96] C. Xiang, P. C. Yong, and L. S. Meng, "Design of multiple-level hybrid

classifier for intrusion detection system using bayesian clustering and decision trees." *Pattern Recognition Letters*, vol. 29, no. 7, pp. 918–924, 2008. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167865508000251. https://doi.org/10.1016/j.patrec.2008.01.008

[97] G. Giacinto, R. Perdisci, M. D. Rio, and F. Roli, "Intrusion detection in computer networks by a modular ensemble of one-class classifiers." *Information Fusion*, vol. 9, no. 1, pp. 69–82, 2008. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1566253506000765. https://doi.org/10.1016/j.inffus.2006.10.002

[98] K. Das, J. Schneider, and D. B. Neill, "Anomaly pattern detection in categorical datasets." in *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2008, pp. 169–176. https://doi.org/10.1145/1401890.1401915

[99] W. Hu, W. Hu, and S. Maybank, "AdaBoost-based algorithm for network intrusion detection." *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 38, no. 2, pp. 577–583, 2008. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-41749107387&doi=10.1109%2fTSMCB.2007.914695&partnerID=40&md5=1622b833beb000ace00407a5bbeff106. https://doi.org/10.1109/TSMCB.2007.914695

[100] A. Tajbakhsh, M. Rahmati, and A. Mirzaei, "Intrusion detection using fuzzy association rules." *Applied Soft Computing*, vol. 9, no. 2, pp. 462–469, 2009. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1568494608000975. https://doi.org/10.1016/j.asoc.2008.06.001

[101] D. Sánchez, M. A. Vila, L. Cerda, and J. M. Serrano, "Association rules applied to credit card fraud detection." *Expert Systems with Applications*, vol. 36, no. 2, Part 2, pp. 3630–3640, 2009. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0957417408001176. https:

//doi.org/10.1016/j.eswa.2008.02.001

[102] K. Shafi and H. A. Abbass, "An adaptive genetic-based signature learning system for intrusion detection." *Expert Systems with Applications*, vol. 36, no. 10, pp. 12 036–12 043, 2009. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0957417409002589. https://doi.org/10.1016/j.eswa.2009.03.036

[103] S. Wu and E. Yen, "Data mining-based intrusion detectors." *Expert Systems with Applications*, vol. 36, no. 3, Part 1, pp. 5605–5612, 2009. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0957417408004089. https://doi.org/10.1016/j.eswa.2008.06.138

[104] T. P. Tran, L. Cao, D. Tran, and C. D. Nguyen, "Novel intrusion detection using probabilistic neural network and adaptive boosting." *International Journal of Computer Science & Information Security*, vol. 6, no. 1, pp. 83–91, 2009.

[105] X. Tong, Z. Wang, and H. Yu, "A research using hybrid RBF/Elman neural networks for intrusion detection system secure model." *Computer Physics Communications*, vol. 180, no. 10, pp. 1795–1801, 2009. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0010465509001519. https://doi.org/10.1016/j.cpc.2009.05.004

[106] W. Lu and H. Tong, "Detecting network anomalies using CUSUM and EM clustering." in *International Symposium on Intelligence Computation and Applications*. Springer, 2009, pp. 297–308. https://doi.org/10.1007/978-3-642-04843-2_32

[107] G. Wang, J. Hao, J. Ma, and L. Huang, "A new approach to intrusion detection using artificial neural networks and fuzzy clustering." *Expert Systems with Applications*, vol. 37, no. 9, pp. 6225–6232, 2010. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0957417410001417. https://doi.org/10.1016/j.eswa.2010.02.102

[108] M. S. Mok, S. Y. Sohn, and Y. H. Ju, "Random effects logistic regression model

for anomaly detection." *Expert Systems with Applications*, vol. 37, no. 10, pp. 7162–7166, 2010. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0957417410002885. https://doi.org/10.1016/j.eswa.2010.04.017

[109] M. M. T. Jawhar and M. Mehrotra, "Design network intrusion detection system using hybrid fuzzy-neural network." *International Journal of Computer Science and Security*, vol. 4, no. 3, pp. 285–294, 2010.

[110] C. Wagner, J. François, and T. Engel, "Machine learning approach for IP-flow record anomaly detection." in *International Conference on Research in Networking*. Springer, 2011, pp. 28–39. https://doi.org/10.1007/978-3-642-20757-0_3

[111] C. M. Rahman, D. M. Farid, and M. Z. Rahman, "Adaptive intrusion detection based on boosting and naive bayesian classifier." *International Journal of Computer Applications*, vol. 24, no. 3, pp. 11–19, 2011.

[112] M. Su, "Real-time anomaly detection systems for denial-of-service attacks by weighted k-nearest-neighbor classifiers." *Expert Systems with Applications*, vol. 38, no. 4, pp. 3492–3498, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0957417410009450. https://doi.org/10.1016/j.eswa.2010.08.137

[113] M. S. Abadeh, H. Mohamadi, and J. Habibi, "Design and analysis of genetic fuzzy systems for intrusion detection in computer networks." *Expert Systems with Applications*, vol. 38, no. 6, pp. 7067–7075, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0957417410013692. https://doi.org/10.1016/j.eswa.2010.12.006

[114] P. Sangkatsanee, N. Wattanapongsakorn, and C. Charnsripinyo, "Practical real-time intrusion detection using machine learning approaches." *Computer Communications*, vol. 34, no. 18, pp. 2227–2235, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S014036641100209X. https://doi.org/10.1016/j.comcom.2011.07.001

[115] S. Lee, G. Kim, and S. Kim, "Self-adaptive and dynamic clustering for online anomaly detection." *Expert Systems with Applications*, vol. 38, no. 12, pp. 14 891–14 898, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0957417411008426. https://doi.org/10.1016/j.eswa.2011.05.058

[116] S. Wang, K. Yan, S. Wang, and C. Liu, "An integrated intrusion detection system for cluster-based wireless sensor networks." *Expert Systems with Applications*, vol. 38, no. 12, pp. 15 234–15 243, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0957417411008608. https://doi.org/10.1016/j.eswa.2011.05.076

[117] Y. Yi, J. Wu, and W. Xu, "Incremental SVM based on reserved set for network intrusion detection." *Expert Systems with Applications*, vol. 38, no. 6, pp. 7698–7707, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0957417410015046. https://doi.org/10.1016/j.eswa.2010.12.141

[118] Z. Muda, W. Yassin, M. N. Sulaiman, and N. I. Udzir, "A k-means and naive bayes learning approach for better intrusion detection." *Information technology journal*, vol. 10, no. 3, pp. 648–655, 2011.

[119] A. S. Aneetha and S. Bose, "The combined approach for anomaly detection using neural networks and clustering techniques." *Computer Science & Engineering*, vol. 2, no. 4, pp. 37–46, 2012.

[120] C. A. Catania, F. Bromberg, and C. G. Garino, "An autonomous labeling approach to support vector machines algorithms for network traffic anomaly detection." *Expert Systems with Applications*, vol. 39, no. 2, pp. 1822–1829, 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0957417411011808. https://doi.org/10.1016/j.eswa.2011.08.068

[121] C. Cheng, W. P. Tay, and G. Huang, "Extreme learning machines for intrusion detection." in *Neural networks (IJCNN), the 2012 international joint conference on*. IEEE, 2012, pp. 1–8. https://doi.org/10.1109/IJCNN.2012.6252449

[122] I. Kang, M. K. Jeong, and D. Kong, "A differentiated one-class classification method with applications to intrusion detection." *Expert Systems with Applications*, vol. 39, no. 4, pp. 3899–3905, 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0957417411009286. https://doi.org/10.1016/j.eswa.2011.06.033

[123] L. Koc, T. A. Mazzuchi, and S. Sarkani, "A network intrusion detection system based on a hidden naïve bayes multiclass classifier." *Expert Systems with Applications*, vol. 39, no. 18, pp. 13 492–13 500, 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0957417412008640. https://doi.org/10.1016/j.eswa.2012.07.009

[124] S. Lin, K. Ying, C. Lee, and Z. Lee, "An intelligent algorithm with feature selection and decision rules applied to anomaly intrusion detection." *Applied Soft Computing*, vol. 12, no. 10, pp. 3285–3290, 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1568494612002402. https://doi.org/10.1016/j.asoc.2012.05.004

[125] S. S. S. Sindhu, S. Geetha, and A. Kannan, "Decision tree based light weight intrusion detection using a wrapper approach." *Expert Systems with Applications*, vol. 39, no. 1, pp. 129–141, 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0957417411009080. https://doi.org/10.1016/j.eswa.2011.06.013

[126] Y. Li, J. Xia, S. Zhang, J. Yan, X. Ai, and K. Dai, "An efficient intrusion detection system based on support vector machines and gradually feature removal method." *Expert Systems with Applications*, vol. 39, no. 1, pp. 424–430, 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0957417411009948. https://doi.org/10.1016/j.eswa.2011.07.032

[127] A. M. Chandrashekhar and K. Raghuveer, "Fortification of hybrid intrusion detection system using variants of neural networks and support vector machines." *International Journal of Network Security and Its Applications*,

vol. 5, no. 1, pp. 71–90, 2013.

[128] D. A. A. Zainaddin and Z. M. Hanapi, "Hybrid of fuzzy clustering neural network over NSL dataset for intrusion detection system." *Journal of Computer Science*, vol. 9, no. 3, pp. 391–403, 2013. https://doi.org/10.3844/jcssp.2013.391.403

[129] M. M. Lisehroodi, Z. Muda, and W. Yassin, "A hybrid framework based on neural network MLP and k-means clustering for intrusion detection system." in *Proceedings of 4th International Conference on Computing and Informatics, ICOCI*, 2013, pp. 305–311.

[130] S. Devaraju and S. Ramakrishnan, "Detection of accuracy for intrusion detection system using neural network classifier." *International Journal of Emerging Technology and Advanced Engineering*, vol. 3, no. 1, pp. 338–345, 2013.

[131] S. Shin, S. Lee, H. Kim, and S. Kim, "Advanced probabilistic approach for network intrusion forecasting and detection." *Expert Systems with Applications*, vol. 40, no. 1, pp. 315–322, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0957417412009128. https://doi.org/10.1016/j.eswa.2012.07.057

[132] W. Yassin, N. I. Udzir, Z. Muda, and M. N. Sulaiman, "Anomaly-based intrusion detection through k-means clustering and naives bayes classification." in *Proceedings of the 4th International Conference on Computing and Informatics, ICOCI 2013*, 2013, pp. 298–303.

[133] Y. Sahin, S. Bulkan, and E. Duman, "A cost-sensitive decision tree approach for fraud detection." *Expert Systems with Applications*, vol. 40, no. 15, pp. 5916–5923, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0957417413003072. https://doi.org/10.1016/j.eswa.2013.05.021

[134] Z. A. Baig, S. M. Sait, and A. Shaheen, "GMDH-based networks for intelligent intrusion detection." *Engineering Applications of Artificial Intelligence*, vol. 26, no. 7, pp. 1731–1740, 2013. [Online]. Available:

http://www.sciencedirect.com/science/article/pii/S095219761300050X. https://doi.org/10.1016/j.engappai.2013.03.008

[135] Z. M. Fadlullah, H. Nishiyama, N. Kato, and M. M. Fouda, "Intrusion detection system (IDS) for combating attacks against cognitive radio networks." *IEEE Network*, vol. 27, no. 3, pp. 51–56, 2013. https://doi.org/10.1109/MNET.2013.6523809

[136] J. Xiang, M. Westerlund, D. Sovilj, and G. Pulkkis, "Using extreme learning machine for intrusion detection in a big data environment." in *Proceedings of the 2014 Workshop on Artificial Intelligent and Security Workshop*. ACM, 2014, pp. 73–82. https://doi.org/10.1145/2666652.2666664

[137] A. K. Shrivas and A. K. Dewangan, "An ensemble model for classification of attacks with feature selection based on KDD99 and NSL-KDD data set." *International Journal of Computer Applications*, vol. 99, no. 15, pp. 8–13, 2014.

[138] G. Kim, S. Lee, and S. Kim, "A novel hybrid intrusion detection method integrating anomaly detection with misuse detection." *Expert Systems with Applications*, vol. 41, no. 4, Part 2, pp. 1690–1700, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0957417413006878. https://doi.org/10.1016/j.eswa.2013.08.066

[139] R. Ranjan and G. Sahoo, "A new clustering approach for anomaly intrusion detection." *International Journal of Data Mining & Knowledge Management Process*, vol. 4, no. 2, pp. 29–38, 2014.

[140] W. Feng, Q. Zhang, G. Hu, and J. X. Huang, "Mining network data for intrusion detection through combining SVMs with ant colony networks." *Future Generation Computer Systems*, vol. 37, pp. 127–140, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X13001416. https://doi.org/10.1016/j.future.2013.06.027

[141] N. A. Seresht and R. Azmi, "MAIS-IDS: A distributed intrusion detection system using multi-agent AIS approach." *Engineering Applications of Artificial*

Intelligence, vol. 35, pp. 286–298, 2014. https://doi.org/10.1016/j.engappai.2014.06.022

[142] A. S. Eesa, Z. Orman, and A. M. A. Brifcani, "A novel feature-selection approach based on the cuttlefish optimization algorithm for intrusion detection systems." *Expert Systems with Applications*, vol. 42, no. 5, pp. 2670–2679, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0957417414006952. https://doi.org/10.1016/j.eswa.2014.11.009

[143] B. W. Masduki, K. Ramli, F. A. Saputra, and D. Sugiarto, "Study on implementation of machine learning methods combination for improving attacks detection accuracy on intrusion detection system (IDS)." in *Quality in Research (QiR), 2015 International Conference on*. IEEE, 2015, pp. 56–64. https://doi.org/10.1109/QiR.2015.7374895

[144] W. Lin, S. Ke, and C. Tsai, "CANN: An intrusion detection system based on combining cluster centers and nearest neighbors." *Knowledge-Based Systems*, vol. 78, pp. 13–21, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0950705115000167. https://doi.org/10.1016/j.knosys.2015.01.009

[145] W. Srimuang and S. Intarasothonchun, "Classification model of network intrusion using weighted extreme learning machine." in *12th international joint conference on Computer science and software engineering (JCSSE)*. IEEE, 2015, pp. 190–194. https://doi.org/10.1109/JCSSE.2015.7219794

[146] E. A. Abas, H. Abdelkader, and A. Keshk, "Artificial immune system based intrusion detection." in *IEEE seventh international conference on intelligent computing and information systems (ICICIS)*. IEEE, 2015, pp. 542–546. https://doi.org/10.1109/IntelCIS.2015.7397274

[147] A. Hadri, K. Chougdali, and R. Touahni, "Intrusion detection system using PCA and fuzzy PCA techniques." in *Advanced Communication Systems and Information Security (ACOSIS), International Conference on*. IEEE, 2016, pp.

1–7. https://doi.org/10.1109/ACOSIS.2016.7843930

[148] B. Subba, S. Biswas, and S. Karmakar, “Enhancing performance of anomaly based intrusion detection systems through dimensionality reduction using principal component analysis.” in *Advanced Networks and Telecommunications Systems (ANTS), 2016 IEEE International Conference on*. IEEE, 2016, pp. 1–6. https://doi.org/10.1109/ANTS.2016.7947776

[149] E. Hodo, X. Bellekens, A. Hamilton, P.-L. Dubouilh, E. Iorkyase, C. Tachtatzis, and R. Atkinson, “Threat analysis of IoT networks using artificial neural network intrusion detection system.” in *2016 International Symposium on Networks, Computers and Communications (IEEE ISNCC’16)*. IEEE, 2016, pp. 1–6. https://doi.org/10.1109/ISNCC.2016.7746067

[150] P. A. Sonewar and S. D. Thosar, “Detection of SQL injection and XSS attacks in three tier web applications.” in *Computing Communication Control and automation (ICCUBEA), 2016 International Conference on*. IEEE, 2016, pp. 1–4. https://doi.org/10.1109/ICCUBEA.2016.7860069

[151] P. Nskh, M. N. Varma, and R. R. Naik, “Principle component analysis based intrusion detection system using support vector machine.” in *Recent Trends in Electronics, Information and Communication Technology (RTEICT), IEEE International Conference on*. IEEE, 2016, pp. 1344–1350. https://doi.org/10.1109/RTEICT.2016.7808050

[152] O. Igbe, I. Darwish, and T. Saadawi, “Distributed network intrusion detection systems: An artificial immune system approach.” in *IEEE First International Conference on Connected Health: Applications, Systems and Engineering Technologies (CHASE)*. IEEE, 2016, pp. 101–106. https://doi.org/10.1109/CHASE.2016.36

[153] G. Osada, K. Omote, and T. Nishide, “Network intrusion detection based on semi-supervised variational auto-encoder.” in *Computer Security – ESORICS 2017*, S. N. Foley, D. Gollmann, and E. Snekkenes, Eds. Cham: Springer

International Publishing, 2017, pp. 344–361. https://doi.org/10.1007/978-3-319-66399-9_19

[154] A. R. Syarif and W. Gata, "Intrusion detection system using hybrid binary PSO and k-nearest neighborhood algorithm." in *11th International Conference on Information and Communication Technology and System (ICTS)*. IEEE, 2017, pp. 181–186. https://doi.org/10.1109/ICTS.2017.8265667

[155] B. Xu, S. Chen, H. Zhang, and T. Wu, "Incremental k-NN SVM method in intrusion detection." in *8th IEEE International Conference on Software Engineering and Service Science (ICSESS)*. IEEE, 2017, pp. 712–717. https://doi.org/10.1109/ICSESS.2017.8343013

[156] C. Tran, T. N. Vo, and T. N. Thinh, "HA-IDS: A heterogeneous anomaly-based intrusion detection system." in *4th NAFOSTED Conference on Information and Computer Science*. IEEE, 2017, pp. 156–161. https://doi.org/10.1109/NAFOSTED.2017.8108056

[157] C. Yin, Y. Zhu, J. Fei, and X. He, "A deep learning approach for intrusion detection using recurrent neural networks." *IEEE Access*, vol. 5, pp. 21 954–21 961, 2017. https://doi.org/10.1109/ACCESS.2017.2762418

[158] D. A. Effendy, K. Kusrini, and S. Sudarmawan, "Classification of intrusion detection system (IDS) based on computer network." in *Proceedings of 2017 2nd International conferences on Information Technology, Information Systems and Electrical Engineering (ICITISEE)*. IEEE, 2017, pp. 90–94. https://doi.org/10.1109/ICITISEE.2017.8285566

[159] E. Hodo, X. Bellekens, E. Iorkyase, A. Hamilton, C. Tachtatzis, and R. Atkinson, "Machine learning approach for detection of nonTor traffic." in *Proceedings of the 12th International Conference on Availability, Reliability and Security*. New York, NY, USA: ACM, 2017, pp. 85:1–85:6. [Online]. Available: http://doi.acm.org/10.1145/3098954.3106068. https://doi.org/10.1145/3098954.3106068

[160] Q. Li, Z. Tan, A. Jamdagni, P. Nanda, X. He, and W. Han, "An intrusion detection system based on polynomial feature correlation analysis." in *2017 IEEE Trustcom/BigDataSE/ICESS*. IEEE, 2017, pp. 978–983. https://doi.org/10.1109/Trustcom/BigDataSE/ICESS.2017.340

[161] S. Zhao, W. Li, T. Zia, and A. Y. Zomaya, "A dimension reduction model and classifier for anomaly-based intrusion detection in internet of things." in *IEEE 15th Intl Conf on Dependable, Autonomic and Secure Computing, 15th Intl Conf on Pervasive Intelligence and Computing, 3rd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech)*. IEEE, 2017, pp. 836–843. https://doi.org/10.1109/DASC-PICom-DataCom-CyberSciTec.2017.141

[162] U. N. Wisesty and Adiwijaya, "Comparative study of conjugate gradient to optimize learning process of neural network for intrusion detection system (IDS)." in *3rd International Conference on Science in Information Technology (ICSITech)*. IEEE, 2017, pp. 459–464. https://doi.org/10.1109/ICSITech.2017.8257156

[163] R. C. Aygun and A. G. Yavuz, "Network anomaly detection with stochastically improved autoencoder based models." in *IEEE 4th International Conference on Cyber Security and Cloud Computing (CSCloud)*. IEEE, 2017, pp. 193–198. https://doi.org/10.1109/CSCloud.2017.39

[164] D. He, X. Chen, D. Zou, L. Pei, and L. Jiang, "An improved kernel clustering algorithm used in computer network intrusion detection." in *Proceedings of 2018 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2018, pp. 1–5. https://doi.org/10.1109/ISCAS.2018.8350994

[165] M. N. Napiah, M. Y. I. Idris, R. Ramli, and I. Ahmedy, "Compression header analyzer intrusion detection system (CHA-IDS) for 6LoWPAN communication protocol." *IEEE Access*, vol. 6, pp. 16 623–16 638, 2018. https://doi.org/10.1109/ACCESS.2018.2798626

[166] M. H. Ali, B. A. D. A. Mohammed, M. A. B. Ismail, and M. F. Zolkipli, "A new intrusion detection system based on fast learning network and particle swarm optimization." *IEEE Access*, vol. 6, pp. 20 255–20 261, 2018. https://doi.org/10.1109/ACCESS.2018.2820092

[167] M. AL-Hawawreh, N. Moustafa, and E. Sitnikova, "Identification of malicious activities in industrial internet of things based on deep learning models." *Journal of Information Security and Applications*, vol. 41, pp. 1–11, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2214212617306002. https://doi.org/10.1016/j.jisa.2018.05.002

[168] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, "A deep learning approach to network intrusion detection." *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 2, no. 1, pp. 41–50, 2018. https://doi.org/10.1109/TETCI.2017.2772792

[169] Q. Zhang, Y. Qu, and A. Deng, "Network intrusion detection using kernel-based fuzzy-rough feature selection." in *2018 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*. IEEE, 2018, pp. 1–6. https://doi.org/10.1109/FUZZ-IEEE.2018.8491578

[170] D. Hooks, X. Yuan, K. Roy, A. Esterline, and J. Hernandez, "Applying artificial immune system for intrusion detection." in *IEEE Fourth International Conference on Big Data Computing Service and Applications (BigDataService)*. IEEE, 2018, pp. 287–292. https://doi.org/10.1109/BigDataService.2018.00051

[171] J. M. Vidal, A. L. S. Orozco, and L. J. G. Villalba, "Adaptive artificial immune networks for mitigating DoS flooding attacks." *Swarm and Evolutionary Computation*, vol. 38, pp. 94–108, 2018. https://doi.org/10.1016/j.swevo.2017.07.002

[172] S. Aljawarneh, M. B. Yassein, and M. Aljundi, "An enhanced J48 classification algorithm for the anomaly intrusion detection systems." *Cluster Computing*, vol. 22, no. 5, pp. 10 549–10 565, 2019. https://doi.org/10.1007/s10586-017-

1109-8

[173] S. Mohammadi, H. Mirvaziri, M. Ghazizadeh-Ahsaee, and H. Karimipour, "Cyber intrusion detection by combined feature selection algorithm." *Journal of information security and applications*, vol. 44, pp. 80–88, 2019. https://doi.org/10.1016/j.jisa.2018.11.007

[174] O. Faker and E. Dogdu, "Intrusion detection using big data and deep learning techniques." in *Proceedings of the 2019 ACM Southeast Conference*. New York, NY, USA: Association for Computing Machinery, 2019, pp. 86–93. https://doi.org/10.1145/3299815.3314439

[175] F. Salo, A. B. Nassif, and A. Essex, "Dimensionality reduction with IG-PCA and ensemble classifier for network intrusion detection." *Computer Networks*, vol. 148, pp. 164–175, 2019. https://doi.org/10.1016/j.comnet.2018.11.010

[176] 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. https://doi.org/10.1109/ACCESS.2019.2895334

[177] J. Ghasemi, J. Esmaily, and R. Moradinezhad, "Intrusion detection system using an optimized kernel extreme learning machine and efficient features." *Sādhanā*, vol. 45, no. 1, pp. 1–9, 2019. https://doi.org/10.1007/s12046-019-1230-x

[178] M. Gharib, B. Mohammadi, S. H. Dastgerdi, and M. Sabokrou, "AutoIDS: Auto-encoder based method for intrusion detection system." *arXiv preprint arXiv:1911.03306*, 2019.

[179] A. Yulianto, P. Sukarno, and N. A. Suwastika, "Improving AdaBoost-based intrusion detection system (IDS) performance on CIC IDS 2017 dataset." in *Journal of Physics: Conference Series*, vol. 1192. IOP Publishing, 2019, p. 012018. https://doi.org/10.1088/1742-6596/1192/1/012018

[180] S. Sen, K. D. Gupta, and M. M. Ahsan, "Leveraging machine learning approach to setup software-defined network (SDN) controller rules during DDoS

attack." in *Proceedings of International Joint Conference on Computational Intelligence*. Springer, 2020, pp. 49–60. https://doi.org/10.1007/978-981-13-7564-4_5

[181] Z. Liu, Y. Zhu, X. Yan, L. Wang, Z. Jiang, and J. Luo, "Deep learning approach for IDS." in *Fourth International Congress on Information and Communication Technology*. Springer, 2020, pp. 471–479. https://doi.org/10.1007/978-981-15-0637-6_40

[182] M. Sarnovsky and J. Paralic, "Hierarchical intrusion detection using machine learning and knowledge model." *Symmetry*, vol. 12, no. 2, p. 203, 2020. https://doi.org/10.3390/sym12020203

[183] M. Eskandari, Z. H. Janjua, M. Vecchio, and F. Antonelli, "Passban IDS: An intelligent anomaly based intrusion detection system for IoT edge devices." *IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 6882–6897, 2020. https://doi.org/10.1109/JIOT.2020.2970501

[184] M. D. Hossain, H. Ochiai, F. Doudou, and Y. Kadobayashi, "SSH and FTP brute-force attacks detection in computer networks: LSTM and machine learning approaches." in *5th International Conference on Computer and Communication Systems (ICCCS)*. IEEE, 2020, pp. 491–497. https://doi.org/10.1109/ICCCS49078.2020.9118459

[185] A. M. A. Tobi and I. Duncan, "KDD 1999 generation faults: A review and analysis." *Journal of Cyber Security Technology*, pp. 1–37, 2018. https://doi.org/10.1080/23742917.2018.1518061

[186] M. V. Mahoney and P. K. Chan, "An analysis of the 1999 DARPA/Lincoln laboratory evaluation data for network anomaly detection." in *Recent Advances in Intrusion Detection*, G. Vigna, C. Kruegel, and E. Jonsson, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 220–237. https://doi.org/10.1007/978-3-540-45248-5_13

[187] J. McHugh, "Testing intrusion detection systems: A critique of the 1998

and 1999 DARPA intrusion detection system evaluations as performed by Lincoln Laboratory." *ACM Transactions on Information and System Security (TISSEC)*, vol. 3, no. 4, pp. 262–294, 2000. [Online]. Available: http://doi.acm.org/10.1145/382912.382923. https://doi.org/10.1145/382912.382923

[188] K. Kendall, "A database of computer attacks for the evaluation of intrusion detection systems." Bachelor of Science, Electrical Engineering and Computer Science, 1999.

[189] K. Siddique, Z. Akhtar, F. A. Khan, and Y. Kim, "KDD Cup 99 data sets: A perspective on the role of data sets in network intrusion detection research." *Computer*, vol. 52, no. 2, pp. 41–51, 2019. https://doi.org/10.1109/MC.2018.2888764

[190] D. Welch and S. Lathrop, "Wireless security threat taxonomy." in *Information Assurance Workshop, 2003. IEEE Systems, Man and Cybernetics Society*. IEEE, 2003, pp. 76–83. https://doi.org/10.1109/SMCSIA.2003.1232404

[191] S. Babar, P. Mahalle, A. Stango, N. Prasad, and R. Prasad, "Proposed security model and threat taxonomy for the internet of things (IoT)." in *International Conference on Network Security and Applications*. Springer, 2010, pp. 420–429. https://doi.org/10.1007/978-3-642-14478-3_42

[192] J. Jung, B. Krishnamurthy, and M. Rabinovich, "Flash crowds and denial of service attacks: Characterization and implications for CDNs and web sites." in *Proceedings of the 11th international conference on World Wide Web*, 2002, pp. 293–304. https://doi.org/10.1145/511446.511485

[193] Neustar, "Cyber threats & trends report: First half 2020." Neustar, Inc., Tech. Rep., 2020. [Online]. Available: https://www.home.neustar/resources/whitepapers/cyber-threats-and-trends-report-2020-first-half

[194] S. McClure, J. Scambray, G. Kurtz, and Kurtz, *Hacking Exposed: Network Security Secrets and Solutions.*, 6th ed. McGraw-Hill, 2009.

[195] PurpleSec, "2020 cyber security statistics: The ultimate list of stats, data &

trends," 08 November 2020, Accessed: 8 February, 2021. [Online]. Available: https://purplesec.us/resources/cyber-security-statistics/

[196] "Malware vs viruses: What's the difference?" February 2018, Accessed: 28 February, 2018. [Online]. Available: https://antivirus.comodo.com/blog/computer-safety/malware-vs-viruses-whats-difference/

[197] B. B. Rad, M. Masrom, and S. Ibrahim, "Camouflage in malware: From encryption to metamorphism." *International Journal of Computer Science and Network Security*, vol. 12, no. 8, pp. 74–83, 2012.

[198] H. S. Galal, Y. B. Mahdy, and M. A. Atiea, "Behavior-based features model for malware detection." *Journal of Computer Virology and Hacking Techniques*, vol. 12, no. 2, pp. 59–67, 2016. https://doi.org/10.1007/s11416-015-0244-0

[199] D. Bruschi, L. Martignoni, and M. Monga, "Code normalization for self-mutating malware." *IEEE Security & Privacy*, vol. 5, no. 2, pp. 46–54, 2007. https://doi.org/10.1109/MSP.2007.31

[200] A. Javed, P. Burnap, and O. Rana, "Prediction of drive-by download attacks on twitter." *Information Processing & Management*, vol. 56, no. 3, pp. 1133–1145, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0306457317305824. https://doi.org/10.1016/j.ipm.2018.02.003

[201] L. Neely, "Threat landscape survey: Users on the front line. sans institute," 2017, Accessed: 18 February, 2018. [Online]. Available: https://www.sans.org/reading-room/whitepapers/threats/paper/37910

[202] SecurityFirst, "The top 9 network security threats of 2019," 24 August 2018, Accessed: May 16, 2019. [Online]. Available: https://securityfirstcorp.com/the-top-9-network-security-threats-of-2019/

[203] P. Parrend, J. Navarro, F. Guigou, A. Deruyver, and P. Collet, "Foundations and applications of artificial intelligence for zero-day and multi-step attack detection." *EURASIP Journal on Information Security*, vol. 2018, no. 1, p. 4, 2018. https://doi.org/10.1186/s13635-018-0074-y

[204] J. Kim, S. Bu, and S. Cho, "Zero-Day malware detection using transferred generative adversarial networks based on deep autoencoders." *Information Sciences*, vol. 460-461, pp. 83–102, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0020025518303475. https://doi.org/10.1016/j.ins.2018.04.092

[205] L. Maglaras, M. A. Ferrag, A. Derhab, M. Mukherjee, H. Janicke, and S. Rallis, "Threats, protection and attribution of cyber attacks on critical infrastructures." *arXiv preprint arXiv:1901.03899*, 2019.

[206] R. Mitchell and I. Chen, "A survey of intrusion detection techniques for cyber-physical systems." *ACM Computing Surveys (CSUR)*, vol. 46, no. 4, pp. 1–29, 2014. https://doi.org/10.1145/2542049

[207] S. Amin, X. Litrico, S. Sastry, and A. M. Bayen, "Cyber security of water SCADA systems—part I: Analysis and experimentation of stealthy deception attacks." *IEEE Transactions on Control Systems Technology*, vol. 21, no. 5, pp. 1963–1970, 2012. https://doi.org/10.1109/TCST.2012.2211873

[208] S. Amin, X. Litrico, S. S. Sastry, and A. M. Bayen, "Cyber security of water SCADA systems—part II: Attack detection using enhanced hydrodynamic models." *IEEE Transactions on Control Systems Technology*, vol. 21, no. 5, pp. 1679–1693, 2013. https://doi.org/10.1109/TCST.2012.2211874

[209] L. Cheng, K. Tian, and D. Yao, "Orpheus: Enforcing cyber-physical execution semantics to defend against data-oriented attacks." in *Proceedings of the 33rd Annual Computer Security Applications Conference*. New York, USA: Association for Computing Machinery, 2017, pp. 315–326. https://doi.org/10.1145/3134600.3134640

[210] A. Mathur, "On the limits of detecting process anomalies in critical infrastructure." in *Proceedings of the 4th ACM Workshop on Cyber-Physical System Security*. New York, USA: Association for Computing Machinery, 2018, p. 1. https://doi.org/10.1145/3198458.3198466

[211] P. Mishra, V. Varadharajan, U. Tupakula, and E. S. Pilli, "A detailed investigation and analysis of using machine learning techniques for intrusion detection." *IEEE Communications Surveys & Tutorials*, vol. 21, no. 1, pp. 686–728, 2019. https://doi.org/10.1109/COMST.2018.2847722

[212] A. Solovev and A. Petrova, "What is the MQTT protocol and why choosing it for IoT devices," 10 September 2020, Accessed: 27 October, 2020. [Online]. Available: https://www.integrasources.com/blog/mqtt-protocol-iot-devices/

[213] V. R. Konasani and S. Kadre, *Machine Learning and Deep Learning Using Python and TensorFlow*. McGraw-Hill Education, 2021. [Online]. Available: https://www.accessengineeringlibrary.com/content/book/9781260462296

[214] D. T. Larose and C. D. Larose, *Discovering Knowledge in Data: An Introduction to Data Mining*. John Wiley & Sons, 2014.

[215] L. Wei-Chao, K. Shih-Wen, and T. Chih-Fong, "Top 10 data mining techniques in business applications: A brief survey." *Kybernetes*, vol. 46, no. 7, pp. 1158–1170, 2017. https://doi.org/10.1108/K-10-2016-0302

[216] D. W. Hosmer Jr, S. Lemeshow, and R. X. Sturdivant, *Applied Logistic Regression*. John Wiley & Sons, 2013, vol. 398.

[217] H. Hegre, "Logistic regression: Binomial, multinomial and ordinal." *Universitetet i Oslo*, pp. 1–35, 23 September 2011. [Online]. Available: https://havardhegre.files.wordpress.com/2014/03/logisticregression2011.pdf

[218] D. Barber, *Bayesian Reasoning and Machine Learning*. Cambridge University Press, 2012.

[219] A. Barone, "Tools for fundemental analysis — conditional probability," 29 August 2020, Accessed: 9 September, 2020. [Online]. Available: https://www.investopedia.com/terms/c/conditional_probability.asp

[220] J. Maillo, I. Triguero, and F. Herrera, "A mapreduce-based k-nearest neighbor approach for big data classification." in *2015 IEEE Trustcom/BigDataSE/ISPA*, vol. 2. IEEE, 2015, pp. 167–172. https://doi.org/10.1109/Trustcom.2015.577

[221] D. Subramanian, "A simple introduction to k-nearest neighbors algorithm," 8 June 2019, Accessed: 27 October, 2020. [Online]. Available: https://towardsdatascience.com/a-simple-introduction-to-k-nearest-neighbors-algorithm-b3519ed98e

[222] J. B. Mitchell, "Machine learning methods in chemoinformatics," *Wiley Interdisciplinary Reviews: Computational Molecular Science*, vol. 4, no. 5, pp. 468–481, 2014.

[223] I. Steinwart and A. Christmann, *Support Vector Machines.* Springer Science & Business Media, 2008.

[224] C. Cortes and V. Vapnik, "Support-vector networks." *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995. https://doi.org/10.1007/BF00994018

[225] A. Ng, "Part V: Support vector machines — CS229 lecture notes." pp. 1–25, 2000. [Online]. Available: http://cs229.stanford.edu/notes/cs229-notes3.pdf

[226] A. Zisserman, "The SVM classifier — C19 machine learning." pp. 1–40, 2015. [Online]. Available: https://www.robots.ox.ac.uk/~az/lectures/ml/lect2.pdf

[227] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python." *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[228] R. Lior, *Data Mining with Decision Trees: Theory and Applications.* World scientific, 2014, vol. 81.

[229] J. Xie, F. R. Yu, T. Huang, R. Xie, J. Liu, C. Wang, and Y. Liu, "A survey of machine learning techniques applied to software defined networking (SDN): Research issues and challenges." *IEEE Communications Surveys & Tutorials*, vol. 21, no. 1, pp. 393–430, 2019. https://doi.org/10.1109/COMST.2018.2866942

[230] G. Van Rossum and F. L. Drake, *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace, 2009.

[231] J. Yin, "Understanding the data splitting functions in scikit-learn," 3 September 2018, Accessed: 21 October, 2020. [Online]. Available: https://medium.com/@julie.yin/understanding-the-data-splitting-functions-in-scikit-learn-9ae4046fbd26

[232] S. Stone and M. Temple, "Radio-frequency-based anomaly detection for programmable logic controllers in the critical infrastructure." *International Journal of Critical Infrastructure Protection*, vol. 5, no. 2, pp. 66–73, 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1874548212000200. https://doi.org/10.1016/j.ijcip.2012.05.001

[233] Statista, "Global number of connected IoT devices 2015-2025," 4 January 2021, Accessed: 20 January, 2021. [Online]. Available: https://www.statista.com/statistics/1101442/iot-number-of-connected-devices-worldwide/

[234] H. Arasteh, V. Hosseinnezhad, V. Loia, A. Tommasetti, O. Troisi, M. Shafie-khah, and P. Siano, "IoT-based smart cities: A survey." in *IEEE 16th International Conference on Environment and Electrical Engineering (EEEIC)*. IEEE, 2016, pp. 1–6. https://doi.org/10.1109/EEEIC.2016.7555867

[235] N. Ahmed, D. De, and I. Hussain, "Internet of things (IoT) for smart precision agriculture and farming in rural areas." *IEEE Internet of Things Journal*, vol. 5, no. 6, pp. 4890–4899, 2018. https://doi.org/10.1109/JIOT.2018.2879579

[236] M. Abdel-Basset, G. Manogaran, and M. Mohamed, "Internet of things (IoT) and its impact on supply chain: A framework for building smart, secure and efficient systems." *Future Generation Computer Systems*, vol. 86, pp. 614–628, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X1830400X. https://doi.org/10.1016/j.future.2018.04.051

[237] Z. Alansari, S. Soomro, M. R. Belgaum, and S. Shamshirband, "The rise of internet of things (IoT) in big healthcare data: Review and open research

issues." in *Progress in Advanced Computing and Intelligent*, K. Saeed, N. Chaki, B. Pati, S. Bakshi, and D. P. Mohapatra, Eds. Singapore: Springer Singapore, Engineering 2018, pp. 675–685. https://doi.org/10.1007/978-981-10-6875-1_66

[238] A. Stanford-Clark and H. L. Truong, "MQTT for sensor networks (MQTT-SN) protocol specification." *International business machines (IBM) Corporation version*, vol. 1, no. 2, pp. 1–28, 2013.

[239] M. S. Harsha, B. M. Bhavani, and K. R. Kundhavai, "Analysis of vulnerabilities in MQTT security using Shodan API and implementation of its countermeasures via authentication and ACLs." in *2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*. IEEE, 2018, pp. 2244–2250. https://doi.org/10.1109/ICACCI.2018.8554472

[240] D. Dinculeană and X. Cheng, "Vulnerabilities and limitations of MQTT protocol used between IoT devices." *Applied Sciences*, vol. 9, no. 5, p. 848, 2019. https://doi.org/10.3390/app9050848

[241] C. Dietz, R. L. Castro, J. Steinberger, C. Wilczak, M. Antzek, A. Sperotto, and A. Pras, "IoT-Botnet detection and isolation by access routers." in *9th International Conference on the Network of the Future (NOF)*. IEEE, 2018, pp. 88–95. https://doi.org/10.1109/NOF.2018.8598138

[242] I. Hafeez, M. Antikainen, A. Y. Ding, and S. Tarkoma, "IoT-KEEPER: Detecting malicious IoT network activity using online traffic analysis at the edge." *IEEE Transactions on Network and Service Management*, vol. 17, no. 1, pp. 45–59, 2020. https://doi.org/10.1109/TNSM.2020.2966951

[243] Vyopta, "What's an acceptable amount of packet loss in 2019?" 19 December 2018, Accessed: 19 November, 2020. [Online]. Available: https://www.vyopta.com/blog/video-conferencing/understanding-packet-loss/

[244] D. Abeles and M. Zioni, "MQTT-PWN, IoT exploitation & recon framework," Accessed: February, 2020. [Online]. Available: https://mqtt-pwn.readthedocs.io/en/latest/index.html

[245] M. Ring, S. Wunderlich, D. Grüdl, D. Landes, and A. Hotho, *A Toolset for Intrusion and Insider Threat Detection.*, ser. Data Analytics and Decision Support for Cybersecurity. Springer, 2017, pp. 3–31. https://doi.org/10.1007/978-3-319-59439-2_1

[246] Y. Roh, G. Heo, and S. E. Whang, "A survey on data collection for machine learning: A Big Data-AI integration perspective." *IEEE Transactions on Knowledge and Data Engineering*, 2019. https://doi.org/10.1109/TKDE.2019.2946162

[247] S. Jain, "NanoNets: How to use deep learning when you have limited data." 30 January 2017, Accessed: 20 November 2018, 2018. [Online]. Available: https://medium.com/nanonets/nanonets-how-to-use-deep-learning-when-you-have-limited-data-f68c0b512cab

[248] S. Roshan, Y. Miche, A. Akusok, and A. Lendasse, "Adaptive and online network intrusion detection system using clustering and extreme learning machines." *Journal of the Franklin Institute*, vol. 355, no. 4, pp. 1752–1779, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0016003217302995. https://doi.org/10.1016/j.jfranklin.2017.06.006

[249] A. L'Heureux, K. Grolinger, H. F. Elyamany, and M. A. M. Capretz, "Machine learning with big data: Challenges and approaches." *IEEE Access*, vol. 5, pp. 7776–7797, 2017. https://doi.org/10.1109/ACCESS.2017.2696365

[250] S. J. Pan and Q. Yang, "A survey on transfer learning." *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 2010. https://doi.org/10.1109/TKDE.2009.191

[251] K. Weiss, T. M. Khoshgoftaar, and D. Wang, "A survey of transfer learning ." *Journal of Big Data*, vol. 3, no. 1, p. 9, 2016. https://doi.org/10.1186/s40537-016-0043-6

[252] Q. Wang, X. Zhao, J. Huang, Y. Feng, Z. Liu, J. Su, Z. Luo, and G. Cheng, "Addressing complexities of machine learning in big data: Principles, trends

and challenges from systematical perspectives." *Preprints*, 2017.

[253] L. D. Nguyen, D. Lin, Z. Lin, and J. Cao, "Deep CNNs for microscopic image classification by exploiting transfer learning and feature concatenation." in *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2018, pp. 1–5. https://doi.org/10.1109/ISCAS.2018.8351550

[254] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, and M. Bernstein, "Imagenet large scale visual recognition challenge." *International journal of computer vision*, vol. 115, no. 3, pp. 211–252, 2015. https://doi.org/10.1007/s11263-015-0816-y

[255] J. Ngiam, D. Peng, V. Vasudevan, S. Kornblith, Q. V. Le, and R. Pang, "Domain adaptive transfer learning with specialist models." *arXiv preprint arXiv:1811.07056*, 2018.

[256] L. Torrey and J. Shavlik, *Transfer Learning.*, ser. Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques. IGI global, 2010, pp. 242–264. https://doi.org/10.4018/978-1-60566-766-9.ch011

[257] X. Sun, J. Dai, P. Liu, A. Singhal, and J. Yen, "Using bayesian networks for probabilistic identification of zero-day attack paths." *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 10, pp. 2506–2521, 2018. https://doi.org/10.1109/TIFS.2018.2821095

[258] S. Chopra, R. Hadsell, and Y. LeCun, "Learning a similarity metric discriminatively, with application to face verification." in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1. IEEE, 2005, pp. 539–546. https://doi.org/10.1109/CVPR.2005.202

[259] L. Fei-Fei, R. Fergus, and P. Perona, "One-Shot learning of object categories." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 4, pp. 594–611, 2006. https://doi.org/10.1109/TPAMI.2006.79

[260] L. Wang, Y. Li, and S. Wang, "Feature learning for One-Shot face recognition."

in *25th IEEE International Conference on Image Processing (ICIP)*. IEEE, 2018, pp. 2386–2390. https://doi.org/10.1109/ICIP.2018.8451464

[261] D. Wu, F. Zhu, and L. Shao, "One Shot learning gesture recognition from RGBD images." in *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*. IEEE, 2012, pp. 7–12. https://doi.org/10.1109/CVPRW.2012.6239179

[262] Y. Yang, I. Saleemi, and M. Shah, "Discovering motion primitives for unsupervised grouping and One-Shot learning of human actions, gestures, and expressions." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 7, pp. 1635–1648, 2013. https://doi.org/10.1109/TPAMI.2012.253

[263] J. Bruce, N. Sünderhauf, P. Mirowski, R. Hadsell, and M. Milford, "One-Shot reinforcement learning for robot navigation with interactive replay." *arXiv preprint arXiv:1711.10137*, 2017.

[264] Z. Zhang and H. Zhao, "One-Shot learning for question-answering in gaokao history challenge." in *Proceedings of the 27th International Conference on Computational Linguistics*, 2018, pp. 449–461.

[265] W. Yin, H. Schütze, B. Xiang, and B. Zhou, "ABCNN: Attention-based convolutional neural network for modeling sentence pairs." *Transactions of the Association of Computational Linguistics*, vol. 4, no. 1, pp. 259–272, 2016. https://doi.org/10.1162/tacl_a_00097

[266] H. Altae-Tran, B. Ramsundar, A. S. Pappu, and V. Pande, "Low data drug discovery with One-Shot learning." *ACS central science*, vol. 3, no. 4, pp. 283–293, 2017. https://doi.org/10.1021/acscentsci.6b00367

[267] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah, "Signature verification using a "siamese" time delay neural network." in *Advances in neural information processing systems*, 1994, pp. 737–744.

[268] G. Koch, R. Zemel, and R. Salakhutdinov, "Siamese neural networks for

One-Shot image recognition." in *Proceedings of the 32nd International Conference on Machine Learning*, 2015.

[269] G. Koch, "Siamese neural networks for One-Shot image recognition." Master of Science, Computer Science, 2015.

[270] Y. Yao, X. Wu, W. Zuo, and D. Zhang, "Learning Siamese network with top-down modulation for visual tracking." in *International Conference on Intelligent Science and Big Data Engineering*. Springer, 2018, pp. 378–388. https://doi.org/10.1007/978-3-030-02698-1_33

[271] R. R. Varior, M. Haloi, and G. Wang, "Gated Siamese convolutional neural network architecture for human re-identification." in *European Conference on Computer Vision*. Springer, 2016, pp. 791–808. https://doi.org/10.1007/978-3-319-46484-8_48

[272] S. Ruder, "An overview of gradient descent optimization algorithms." *arXiv preprint arXiv:1609.04747*, 2016.

[273] U. Shaham and R. R. Lederman, "Learning by coincidence: Siamese networks and common variable learning." *Pattern Recognition*, vol. 74, pp. 52–63, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0031320317303588. https://doi.org/10.1016/j.patcog.2017.09.015

[274] R. Hadsell, S. Chopra, and Y. LeCun, "Dimensionality reduction by learning an invariant mapping." in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, vol. 2. IEEE, 2006, pp. 1735–1742. https://doi.org/10.1109/CVPR.2006.100

[275] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization." *The Journal of Machine Learning Research*, vol. 13, no. 1, pp. 281–305, 2012.

[276] B. M. Lake, R. Salakhutdinov, and J. B. Tenenbaum, "The omniglot challenge: A 3-year progress report." *Current Opinion in Behavioral Sciences*, vol. 29, pp. 97–104, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2352154619300051. https://doi.org/10.1016/j.cobeha.2019.04.007

[277] S. Pang, S. Qiao, T. Song, J. Zhao, and P. Zheng, "An improved convolutional network architecture based on residual modeling for person re-identification in edge computing." pp. 106 748–106 759, 2019. https://doi.org/10.1109/ACCESS.2019.2933364

[278] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity." *The Bulletin of Mathematical Biophysics*, vol. 5, no. 4, pp. 115–133, 1943. https://doi.org/10.1007/BF02478259

[279] J. L. McClelland and D. E. Rumelhart, *A Distributed Model of Human Learning and Memory.*, ser. Parallel distributed processing: Explorations in the microstructure of cognition, Vol. 2: Psychological and biological models. Cambridge, MA, USA: MIT Press, 1986, pp. 170–215.

[280] S. Sharma, S. Sharma, and A. Athaiya, "Activation functions in neural networks." *International Journal of Engineering Applied Sciences and Technology*, vol. 4, no. 12, pp. 310–316, 2020.

[281] Algorithmia, "Introduction to optimizers." 7 May 2018, Accessed: Jul 16, 2021, 2021. [Online]. Available: https://algorithmia.com/blog/introduction-to-optimizers

[282] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[283] DeepAI, "Weight (artificial neural network)," 17 May 2019, Accessed: 4 November, 2020. [Online]. Available: https://deepai.org/machine-learning-glossary-and-terms/weight-artificial-neural-network

[284] T. Yiu, "Understanding neural networks," 2 June 2019, Accessed: 4 November, 2020. [Online]. Available: https://towardsdatascience.com/understanding-neural-networks-19020b758230

[285] "KDD Cup 1999 data," Accessed: 12 July, 2018. [Online]. Available: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

[286] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis

of the KDD CUP 99 data set." in *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*. IEEE, 2009, pp. 1–6. https://doi.org/10.1109/CISDA.2009.5356528

[287] D. Preethi and N. Khare, "Sparse auto encoder driven support vector regression based deep learning model for predicting network intrusions." *Peer-to-Peer Networking and Applications*, pp. 1–11, 2020. https://doi.org/10.1007/s12083-020-00986-3

[288] K. Martin, N. Wiratunga, S. Massie, and J. Clos, "Informed pair selection for self-paced metric learning in Siamese neural networks." in *Artificial Intelligence XX*, M. Bramer and M. Petridis, Eds. Cham: Springer International Publishing, XV 2018, pp. 34–49. https://doi.org/10.1007/978-3-030-04191-5_3

[289] J. Brownlee, "How to configure the number of layers and nodes in a neural network," 27 July 2018, Accessed: 25 September, 2020. [Online]. Available: https://machinelearningmastery.com/how-to-configure-the-number-of-layers-and-nodes-in-a-neural-network/

[290] S. Sayad, "ZeroR," Accessed: 8 December, 2019. [Online]. Available: https://www.saedsayad.com/zeror.htm

[291] N. Japkowicz and S. Stephen, "The class imbalance problem: A systematic study." *Intelligent data analysis*, vol. 6, no. 5, pp. 429–449, 2002. https://doi.org/10.3233/IDA-2002-6504

[292] J. M. Johnson and T. M. Khoshgoftaar, "Survey on deep learning with class imbalance." *Journal of Big Data*, vol. 6, no. 1, p. 27, 2019. https://doi.org/10.1186/s40537-019-0192-5

[293] P. Illy, G. Kaddoum, C. M. Moreira, K. Kaur, and S. Garg, "Securing fog-to-things environment using intrusion detection system based on ensemble learning." in *2019 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, 2019, pp. 1–7. https://doi.org/10.1109/WCNC.2019.8885534

[294] Y. Li, Y. Xu, Z. Liu, H. Hou, Y. Zheng, Y. Xin, Y. Zhao, and L. Cui,

"Robust detection for network intrusion of industrial IoT based on multi-CNN fusion." *Measurement*, vol. 154, p. 107450, 2020. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S026322411931317X. https://doi.org/10.1016/j.measurement.2019.107450

[295] N. Kaloudi and J. Li, "The AI-based cyber threat landscape: A survey." *ACM Computing Surveys (CSUR)*, vol. 53, no. 1, pp. 1–34, 2020. https://doi.org/10.1145/3372823

[296] T. T. Nguyen and V. J. Reddi, "Deep reinforcement learning for cyber security." *arXiv preprint arXiv:1906.05799*, 2019.

[297] C. Chapman, *Chapter 1 - Introduction to Practical Security and Performance Testing.*, ser. Network Performance and Security. Boston: Syngress, 2016, pp. 1–14. [Online]. Available: http://www.sciencedirect.com/science/article/pii/B9780128035849000019. https://doi.org/10.1016/B978-0-12-803584-9.00001-9

[298] 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*. ACM, 2012, pp. 833–844. https://doi.org/10.1145/2382196.2382284

[299] K. Metrick, P. Najafi, and J. Semrau, "Zero-Day exploitation increasingly demonstrates access to money, rather than skill — intelligence for vulnerability management, part one — FireEye inc," April 2020, Accessed: 24 June, 2020. [Online]. Available: https://www.fireeye.com/blog/threat-research/2020/04/zero-day-exploitation-demonstrates-access-to-money-not-skill.html

[300] F. Iglesias, A. Hartl, T. Zseby, and A. Zimek, "Are network attacks outliers? A study of space representations and unsupervised algorithms." in *Machine Learning and Knowledge Discovery in Databases*, P. Cellier and K. Driessens, Eds. Cham: Springer International Publishing, 2020, pp. 159–175. https://doi.org/10.1007/978-3-030-43887-6_13

[301] Cisco, "Cisco 2017 annual cyber security report," 2017, Accessed: 20 July, 2020. [Online]. Available: https://www.grouppbs.com/wp-content/uploads/2017/02/Cisco_2017_ACR_PDF.pdf

[302] Cisco Secure, "Cisco cybersecurity report series 2020: CISO benchmark study. securing what's now and what's next. 20 cybersecurity considerations for 2020." Cisco, Tech. Rep., February 2020. [Online]. Available: 'https://www.cisco.com/c/dam/en/us/products/collateral/security/2020-ciso-benchmark-cybersecurity-series-feb-2020.pdf

[303] S. Alam, S. K. Sonbhadra, S. Agarwal, and P. Nagabhushan, "One-class support vector classifiers: A survey." *Knowledge-Based Systems*, vol. 196, p. 105754, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950705120301647. https://doi.org/10.1016/j.knosys.2020.105754

[304] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation." California Univ San Diego La Jolla Inst for Cognitive Science, Tech. Rep., 1985.

[305] M. Stewart, "Comprehensive introduction to autoencoder," April 2019, Accessed: 21 July, 2020. [Online]. Available: https://towardsdatascience.com/generating-images-with-autoencoders-77fd3a8dd368

[306] K. Pawar and V. Z. Attar, *Assessment of Autoencoder Architectures for Data Representation.*, ser. Deep Learning: Concepts and Architectures. Cham: Springer International Publishing, 2020, pp. 101–132. [Online]. Available: https://link.springer.com/chapter/10.1007%2F978-3-030-31756-0_4. https://doi.org/10.1007/978-3-030-31756-0_4

[307] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks." *Science*, vol. 313, no. 5786, pp. 504–507, 2006. https://doi.org/10.1126/science.1127647

[308] J. Zabalza, J. Ren, J. Zheng, H. Zhao, C. Qing, Z. Yang, P. Du, and S. Marshall, "Novel segmented stacked autoencoder for effective dimensionality reduction

and feature extraction in hyperspectral imaging." *Neurocomputing*, vol. 185, pp. 1–10, 2016. https://doi.org/10.1016/j.neucom.2015.11.044

[309] D. Barber, "Implicit representation networks." University College London, Department of Computer Science, Tech. Rep., 2014.

[310] E. Plaut, "From principal subspaces to principal components with linear autoencoders," *arXiv preprint arXiv:1804.10253*, 2018.

[311] C. Liou, W. Cheng, J. Liou, and D. Liou, "Autoencoder for words." *Neurocomputing*, vol. 139, pp. 84–96, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0925231214003658. https://doi.org/10.1016/j.neucom.2013.09.055

[312] L. Theis, W. Shi, A. Cunningham, and F. Huszár, "Lossy image compression with compressive autoencoders." in *International Conference on Learning Representations (ICLR 2017)*, 2017. https://doi.org/10.17863/CAM.51995

[313] C. Zhou and R. C. Paffenroth, "Anomaly detection with robust deep autoencoders." in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York, USA: Association for Computing Machinery, 2017, pp. 665–674. https://doi.org/10.1145/3097983.3098052

[314] A. Creswell and A. A. Bharath, "Denoising adversarial autoencoders." *IEEE transactions on neural networks and learning systems*, vol. 30, no. 4, pp. 968–984, 2018. https://doi.org/10.1109/TNNLS.2018.2852738

[315] B. Schölkopf, R. C. Williamson, A. J. Smola, J. Shawe-Taylor, and J. C. Platt, "Support vector method for novelty detection." in *Advances in neural information processing systems*, 2000, pp. 582–588.

[316] B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson, "Estimating the support of a high-dimensional distribution." *Neural Computation*, vol. 13, no. 7, pp. 1443–1471, 2001. https://doi.org/10.1162/089976601750264965

[317] D. M. J. Tax and R. P. W. Duin, "Support vector data description." *Machine Learning*, vol. 54, no. 1, pp. 45–66, 2004. https://doi.org/10.1023/B:MACH.0000008084.60811.49

[318] S. Wang, Q. Liu, E. Zhu, F. Porikli, and J. Yin, "Hyperparameter selection of one-class support vector machine by self-adaptive data shifting." *Pattern Recognition*, vol. 74, pp. 198–211, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0031320317303564. https://doi.org/10.1016/j.patcog.2017.09.012

[319] Scikit-learn Developers, "One-class SVM with non-linear kernel (RBF)," 2007, Accessed: 04 December, 2020. [Online]. Available: https://scikit-learn.org/stable/auto_examples/svm/plot_oneclass.html

[320] Y. N. Kunang, S. Nurmaini, D. Stiawan, A. Zarkasi, Firdaus, and Jasmir, "Automatic features extraction using autoencoder in intrusion detection system." in *2018 International Conference on Electrical Engineering and Computer Science (ICECOS)*. IEEE, 2018, pp. 219–224. https://doi.org/10.1109/ICECOS.2018.8605181

[321] Z. Kherlenchimeg and N. Nakaya, "Network intrusion classifier using autoencoder with recurrent neural network." in *Proceedings of The Fourth International Conference on Electronics and Software Science (ICESS2018), Japan*, Japan, 2018, pp. 94–100.

[322] R. A. Shaikh and S. V. Shashikala, "An autoencoder and LSTM based intrusion detection approach against denial of service attacks." in *1st International Conference on Advances in Information Technology (ICAIT)*. IEEE, 2019, pp. 406–410. https://doi.org/10.1109/ICAIT47043.2019.8987336

[323] B. Abolhasanzadeh, "Nonlinear dimensionality reduction for intrusion detection using auto-encoder bottleneck features." in *7th Conference on Information and Knowledge Technology (IKT)*. IEEE, 2015, pp. 1–5. https://doi.org/10.1109/IKT.2015.7288799

[324] F. Farahnakian and J. Heikkonen, "A deep auto-encoder based approach for intrusion detection system." in *20th International Conference on Advanced Communication Technology (ICACT)*. IEEE, 2018, p. 1. https://doi.org/10.23919/ICACT.2018.8323687

[325] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "A detailed analysis of the CICIDS2017 data set." in *Information Systems Security and Privacy*, P. Mori, S. Furnell, and O. Camp, Eds. Cham: Springer International Publishing, 2019, pp. 172–188. https://doi.org/10.1007/978-3-030-25109-3_9

[326] S. Guggisberg, "How to split a dataframe into train and test set with Python," May 2020, Accessed: 17 August, 2020. [Online]. Available: https://towardsdatascience.com/how-to-split-a-dataframe-into-train-and-test-set-with-python-eaa1630ca7b3

[327] A. V. Srinivasan, "Why exclude highly correlated features when building regression model?" 23 August 2019, Accessed: 28 October, 2020. [Online]. Available: https://towardsdatascience.com/why-exclude-highly-correlated-features-when-building-regression-model-34d77a90ea8e

[328] R. Vishal, "Feature selection — correlation and P-value," 11 September 2018, Accessed: 8 March, 2021. [Online]. Available: https://towardsdatascience.com/feature-selection-correlation-and-p-value-da8921bfb3cf

[329] P. Liashchynskyi and P. Liashchynskyi, "Grid search, random search, genetic algorithm: A big comparison for NAS." *arXiv preprint arXiv:1912.06059*, 2019.

[330] P. Chen, C. Lin, and B. Schölkopf, "A tutorial on ν-support vector machines." *Applied Stochastic Models in Business and Industry*, vol. 21, no. 2, pp. 111–136, 2005. https://doi.org/10.1002/asmb.537

[331] K. V. Ravi, V. V. Kumari, and S. S. Kumar, "A survey of feature selection techniques in intrusion detection system: A soft computing perspective." in *Progress in Computing*, P. K. Pattnaik, S. S. Rautaray, H. Das, and J. Nayak, Eds. Singapore: Springer Singapore, Analytics and Networking 2018, pp. 785–793.

https://doi.org/10.1007/978-981-10-7871-2_75

[332] H. Alaidaros, M. Mahmuddin, and A. A. Mazari, "An overview of flow-based and packet-based intrusion detection performance in high speed networks." in *Proceedings of the International Arab Conference on Information Technology*, 2011, pp. 1–9.

[333] F. Y. Rashid, "Encryption, privacy in the internet trends report," June 2019, Accessed: 14 September, 2020. [Online]. Available: https://duo.com/decipher/encryption-privacy-in-the-internet-trends-report

[334] R. Panigrahi and S. Borah, "A detailed analysis of CICIDS2017 dataset for designing intrusion detection systems." *International Journal of Engineering & Technology*, vol. 7, no. 3.24, pp. 479–482, 2018.

[335] D. Aksu and M. A. Aydin, "Detecting port scan attempts with comparative analysis of deep learning and support vector machine algorithms." in *2018 International Congress on Big Data, Deep Learning and Fighting Cyber Terrorism (IBIGDELFT)*. IEEE, 2018, pp. 77–80. https://doi.org/10.1109/IBIGDELFT.2018.8625370

[336] A. A. Abdulrahman and M. K. Ibrahem, "Evaluation of DDoS attacks detection in a new intrusion dataset based on classification algorithms." *Iraqi Journal of Information & Communications Technology*, vol. 1, no. 3, pp. 49–55, 2018.

[337] M. Kuhn and K. Johnson, *Recursive Feature Elimination.*, ser. Feature Engineering and Selection: A Practical Approach for Predictive Models. Taylor & Francis Group, 2019, ch. 11.3. [Online]. Available: https://bookdown.org/max/FES/recursive-feature-elimination.html

[338] J. Brownlee, "Recursive feature elimination (RFE) for feature selection in Python," 25 May 2020, Accessed: 5 November, 2020. [Online]. Available: https://machinelearningmastery.com/rfe-feature-selection-in-python/

[339] OffSec Services, "SlowHTTPTest — penetration testing tools," 2020,

Accessed: 29 July, 2020. [Online]. Available: https://tools.kali.org/stress-testing/slowhttptest

[340] M. Alrowaily, F. Alenezi, and Z. Lu, "Effectiveness of machine learning based intrusion detection systems." in *Security, Privacy, and Anonymity in Computation, Communication, and Storage*, G. Wang, J. Feng, M. Z. A. Bhuiyan, and R. Lu, Eds. Cham: Springer International Publishing, 2019, pp. 277–288. https://doi.org/10.1007/978-3-030-24907-6_21

[341] R. Vijayanand, D. Devaraj, and B. Kannapiran, "Intrusion detection system for wireless mesh network using multiple support vector machine classifiers with genetic-algorithm-based feature selection." *Computers & Security*, vol. 77, pp. 304–314, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167404818303766. https://doi.org/10.1016/j.cose.2018.04.010

[342] M. Bello, G. Nápoles, R. Morera, K. Vanhoof, and R. Bello, "Outliers detection in multi-label datasets," in *Advances in Soft Computing*, L. Martínez-Villaseñor, O. Herrera-Alcántara, H. Ponce, and F. A. Castro-Espinoza, Eds. Cham: Springer International Publishing, 2020, pp. 65–75. https://doi.org/10.1016/j.ins.2020.06.017

[343] E. Hoffer and N. Ailon, "Deep metric learning using triplet network." in *Similarity-Based Pattern Recognition*, A. Feragen, M. Pelillo, and M. Loog, Eds. Cham: Springer International Publishing, 2015, pp. 84–92. https://doi.org/10.1007/978-3-319-24261-3_7

[344] Y. Jun, C. Zhu, J. Zhang, Q. Huang, and D. Tao, "Spatial pyramid-enhanced NetVLAD with weighted triplet loss for place recognition." *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 2, pp. 661–674, 2020. https://doi.org/10.1109/TNNLS.2019.2908982

[345] N. Hoque, M. H. Bhuyan, R. C. Baishya, D. K. Bhattacharyya, and J. K. Kalita, "Network attacks: Taxonomy, tools and systems." *Journal of Network and Computer Applications*, vol. 40, pp. 307–324, 2014. [Online]. Available:

http://www.sciencedirect.com/science/article/pii/S1084804513001756. https://doi.org/10.1016/j.jnca.2013.08.001

Appendices

Appendix A

IDS Datasets Remarks

Section 2.4 discusses different IDS datasets and the attacks included in each. Table A.1 summarises the different institutes that contributes to the generation of the discussed datasets alongside the details of the attack types they cover (if available).

Table A.1
IDS Datasets Remarks

Dataset Name	Institute	Attacks Remarks
General-Purpose Networks		
ADFA-IDS 2017 [65, 66]	Australian Defense Force Academy	Brute-force (FTP and SSH), U2R, Webshell, and remote code execution
Booters [69]	University of Twente, SURFnet, Federal University of Rio Grande do Sul	9 DDoS attacks
Botnet dataset [71]	CIC	7 Botnet types in training set and 16 in test set
CAIDA DDoS [76]	CAIDA	1 hour of DDoS attack divided into 5-minute PCAP files
CIC DoS dataset [64]	CIC	8 DoS attack traces
CICIDS2017 [63]		Brute-force (FTP and SSH), 4 DoS types, DDoS, Heartbleed, Web Attacks, Infiltration Dropbox Download and Cool disk, Botnet, and PortScan
CICIDS2018 [62]		

Table A.1 continued		
Dataset Name	**Institute**	**Attacks Remarks**
CTU-13 [73]	CTU University	13 captures of botnet samples
DARPA [79]	MIT Lincoln Laboratory	17 DoS, 12 U2R, 15 R2L, and 10 Probing
DDoSTB [68]	Punjab Technical University & SBS State Technical Campus	DDoS Testbed using emulated and real nodes
ISCXIDS2012 [74]	CIC	HTTP, SMTP, SSH, IMAP, POP3, and FTP Traffic
KDD-99 [78]	University of California	Covers 24 training attack types and 14 additional types in the test data
TUIDS [70]	Tezpur University	(1) TUIDS IDS dataset. (2) TUIDS Scan dataset. (3) TUIDS DDoS dataset (22 DDoS attack types)
NSL-KDD [77]	CIC	Improvement of KDD'99 dataset
STA2018 [72]	University of St Andrews	Transformation of UNB ISCX (contains 550 features)
Unified Network Dataset [67]	Los Alamos National Laboratory	90 days of Network and Host logs
Waikato [75]	RIPE Network Coordination Centre	UDP traffic only
Special-Purpose Networks		
4SICS ICS [85]	Netresec	-
Anomalies Water System [81]	French Naval Academy	15 different real situations covering cyber attacks (DoS & Spoofing), breakdown (Sensor Failure & Wrong connection), sabotage (Blocked Measures & Floating Objects)
Bot-IoT [80]	The centre of UNSW Canberra Cyber	Attacks include DoS/DDoS, OS and Service Scan, Keylogging and Data Exfiltration
IoT devices captures [82]	Aalto University	represents the data of 31 smart home IoT devices of 27 different types
Tor-nonTor dataset [83]	CIC	7 traffic categories (Browsing, Email, Chat, Audio/Video-Streaming, FTP, VoIP, P2P)

Table A.1 continued		
Dataset Name	**Institute**	**Attacks Remarks**
VPN-nonVPN dataset [84]		14 traffic categories (VPN-VOIP, VPN-P2P, etc.) covering Browsing, Email, Chat, Streaming, File Transfer, VoIP, TraP2P

Appendix B

Attack Tools

Section 3.2 provides a taxonomy of networking threats. The analysis of attacks considered by recent IDS, with respect to the taxonomy, demonstrates the lack of attack representation is IDS datasets. In order to represent these threats, various tools [194, 345] can be used to initiate different attacks. Figure B.1 lists the main tools classified by the attacks they are used to initiate. This can be used by researchers when generating new IDS datasets. For a specific attack, the associated tools are used to launch it, which leads to collecting the relevant traffic data. For example, for impersonation attacks, Caffe-Latte, Wep0ff, and Cain and Abel are tools to consider. Nmap and Netcat are used for scanning and so on.

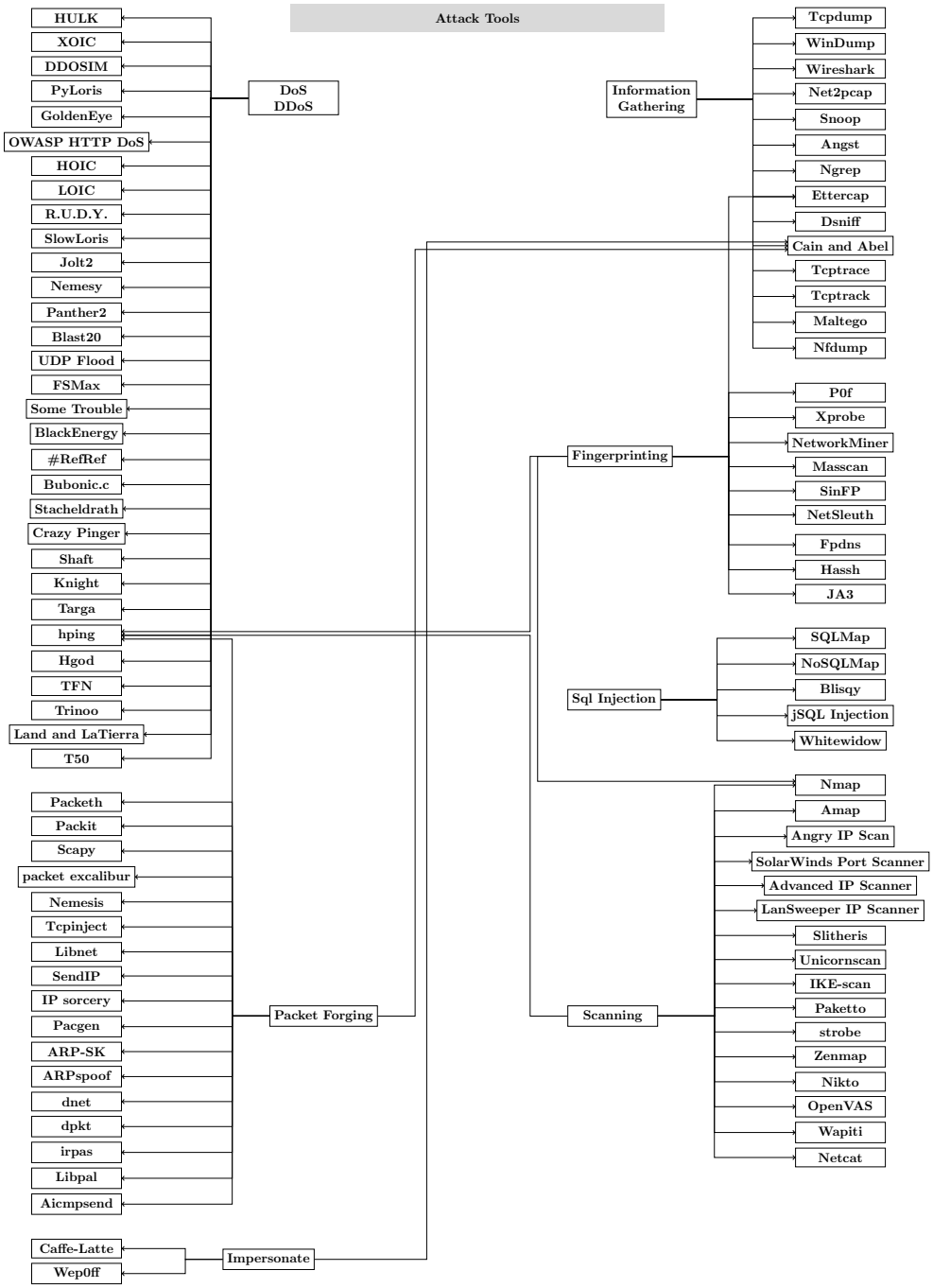


Figure B.1
Attacking Tools

Appendix C

SCADA Dataset Classification Results Tables

Section 4.4 discusses the results of the SCADA scenarios classification. The full results are listed here for completeness and reproducibility.

Table C.1
SCADA Results: Scenarios Classification (5-fold cross-validation) - LR

Class (Scenario)	Recall	Precision	F1-Score
LR			
Normal	7.25%	83.05%	13.31%
Plastic bag	6.05%	51.75%	10.78%
Blocked measure 1	0%	0%	0%
Blocked measure 2	100%	100%	100%
Floating objects in main tank (2 objects)	81.74%	48.39%	60.76%
Floating objects in main tank (7 objects)	95.5%	84.71%	89.76%
Humidity	0%	0%	0%
Sensor Failure	86.89%	54.59%	67.05%
DoS	100%	100%	100%
Spoofing	100%	55.33%	71.24%
Wrong connection	44.1%	68.33%	53.58%
Person hitting the tanks (low intensity)	100%	100%	100%
Person hitting the tanks (med intensity)	100%	100%	100%
Person hitting the tanks (high intensity)	92.85%	85.19%	88.77%
Accuracy	60.95%	63.37%	63.37%

Table C.2
SCADA Results: Scenarios Classification (5-fold cross-validation) - NB

Class (Scenario)	Recall	Precision	F1-Score
NB			
Normal	21.54%	45.55%	29.23%
Plastic bag	0%	0%	0%
Blocked measure 1	100%	100%	100%
Blocked measure 2	100%	100%	100%
Floating objects in main tank (2 objects)	29.28%	35.69%	32.09%
Floating objects in main tank (7 objects)	96.59%	33.55%	49.79%
Humidity	100%	100%	100%
Sensor Failure	3.85%	100%	7.41%
DoS	99.68%	100%	99.84%
Spoofing	100%	44.68%	61.75%
Wrong connection	49.15%	20.44%	28.86%
Person hitting the tanks (low intensity)	100%	100%	100%
Person hitting the tanks (med intensity)	100%	100%	100%
Person hitting the tanks (high intensity)	100%	53.46%	68.28%
Accuracy	41.12%	54.4%	54.4%

Table C.3
SCADA Results: Scenarios Classification (5-fold cross-validation) - k-NN

Class (Scenario)	Recall	Precision	F1-Score
k-NN			
Normal	76.71%	74.11%	75.37%
Plastic bag	75.91%	69.04%	72.3%
Blocked measure 1	100%	100%	100%
Blocked measure 2	100%	100%	100%
Floating objects in main tank (2 objects)	70.14%	75.98%	72.91%
Floating objects in main tank (7 objects)	96.73%	95.33%	96.01%
Humidity	100%	100%	100%
Sensor Failure	76.23%	75.12%	75.66%
DoS	100%	100%	100%
Spoofing	89.7%	94.58%	92.07%
Wrong connection	75.1%	81.95%	78.36%
Person hitting the tanks (low intensity)	100%	100%	100%
Person hitting the tanks (med intensity)	100%	100%	100%
Person hitting the tanks (high intensity)	97.26%	100%	98.58%
Accuracy	81.19%	81.51%	81.51%

Table C.4
SCADA Results: Scenarios Classification (5-fold cross-validation) - SVM

Class (Scenario)	Recall	Precision	F1-Score
SVM			
Normal	23.29%	92.35%	37.18%
Plastic bag	77.26%	56.97%	65.53%
Blocked measure 1	100%	100%	100%
Blocked measure 2	100%	100%	100%
Floating objects in main tank (2 objects)	29.28%	92.38%	44.37%
Floating objects in main tank (7 objects)	95.91%	86.08%	90.72%
Humidity	35.48%	47.41%	34.11%
Sensor Failure	89.49%	54.79%	67.95%
DoS	100%	100%	100%
Spoofing	88.86%	95.5%	92.05%
Wrong connection	44.25%	87.94%	58.76%
Person hitting the tanks (low intensity)	100%	100%	100%
Person hitting the tanks (med intensity)	100%	100%	100%
Person hitting the tanks (high intensity)	100%	100%	100%
Accuracy	70.68%	77.94%	77.94%

Table C.5
SCADA Results: Scenarios Classification (5-fold cross-validation) - Kernel SVM

Class (Scenario)	Recall	Precision	F1-Score
Kernel SVM			
Normal	54.07%	78%	63.78%
Plastic bag	77.76%	58.65%	66.86%
Blocked measure 1	100%	100%	100%
Blocked measure 2	100%	100%	100%
Floating objects in main tank (2 objects)	29.28%	92.1%	44.33%
Floating objects in main tank (7 objects)	95.09%	86.51%	90.59%
Humidity	100%	100%	100%
Sensor Failure	87.07%	61.84%	72.31%
DoS	100%	100%	100%
Spoofing	87.35%	90.81%	89.03%
Wrong connection	45.9%	97.38%	62.33%
Person hitting the tanks (low intensity)	100%	100%	100%
Person hitting the tanks (med intensity)	100%	100%	100%
Person hitting the tanks (high intensity)	98.64%	100%	99.31%
Accuracy	74.78%	79.26%	79.26%

Table C.6
SCADA Results: Scenarios Classification (5-fold cross-validation) - DT

Class (Scenario)	Recall	Precision	F1-Score
DT			
Normal	74.91%	74.72%	74.8%
Plastic bag	70.02%	70.38%	70.18%
Blocked measure 1	100%	100%	100%
Blocked measure 2	100%	100%	100%
Floating objects in main tank (2 objects)	75.66%	74.72%	75.12%
Floating objects in main tank (7 objects)	96.05%	96.32%	96.18%
Humidity	100%	100%	100%
Sensor Failure	74.01%	74.57%	74.28%
DoS	100%	99.06%	99.52%
Spoofing	91.35%	91.11%	91.23%
Wrong connection	77%	76.46%	76.67%
Person hitting the tanks (low intensity)	100%	100%	100%
Person hitting the tanks (med intensity)	98.93%	100%	99.45%
Person hitting the tanks (high intensity)	99.66%	100%	99.83%
Accuracy	80.38%	80.41%	80.41%

Table C.7
SCADA Results: Scenarios Classification (5-fold cross-validation) - RF

Class (Scenario)	Recall	Precision	F1-Score
RF			
Normal	76.21%	74.27%	75.22%
Plastic bag	72.81%	70.79%	71.76%
Blocked measure 1	100%	99.57%	99.78%
Blocked measure 2	99.31%	100%	99.65%
Floating objects in main tank (2 objects)	77.53%	74.02%	75.67%
Floating objects in main tank (7 objects)	97.82%	95.65%	96.71%
Humidity	100%	100%	100%
Sensor Failure	76.05%	74.95%	75.48%
DoS	99.68%	100%	99.84%
Spoofing	90.8%	93.62%	92.19%
Wrong connection	74.1%	80.89%	77.31%
Person hitting the tanks (low intensity)	100%	100%	100%
Person hitting the tanks (med intensity)	100%	99.66%	99.83%
Person hitting the tanks (high intensity)	99.66%	100%	99.83%
Accuracy	81.19%	81.35%	81.35%

Appendix D

Siamese One-Shot Learning Results Tables

In Section 5.6, the proposed Siamese network model is evaluated on classifying a new cyber attack class without the need for retraining. An attack class is excluded from training, one at a time, and used to mimic a new cyber attack with a few instances available. For completeness and transparency, the following subsections list the confusion matrices of the different attacks in the four datasets that are used for evaluation. Furthermore, the results tables list the performance with different number of pairs (j).

D.1 SCADA Dataset

Table D.1
Siamese Network: SCADA One-Shot Confusion Matrix ($j = 5$) (Wrong Connection excluded from Training)

	Predicted Class														
Correct	Normal (S1)	Blocked measure 1 (S2)	Blocked measure 2 (S3)	DoS (S4)	Humidity (S5)	2 Floating objects (S6)	7 Floating objects (S7)	Person hitting high intensity (S8)	Person hitting med intensity (S9)	Person hitting low intensity (S10)	Plastic bag (S11)	Sensor failure (S12)	Spoofing (S13)	Wrong connection (S14)	Overall
S1	600 **(30%)**	150 *(7.5%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	112 *(5.6%)*	491 *(24.55%)*	0 *(0%)*	0 *(0%)*	394 *(19.7%)*	41 *(2.05%)*	163 *(8.15%)*	49 *(2.45%)*	
S2	0 *(0%)*	2000 **(100%)**	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	
S3	0 *(0%)*	0 *(0%)*	2000 **(100%)**	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	
S4	0 *(0%)*	0 *(0%)*	0 *(0%)*	2000 **(100%)**	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	67.87%
S5	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	2000 **(100%)**	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	
S6	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	1201 *(60.05%)*	665 **(33.25%)**	5 *(0.25%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	27 *(1.35%)*	40 *(2%)*	62 *(3.1%)*	
S7	6 *(0.3%)*	535 *(26.75%)*	141 *(7.05%)*	0 *(0%)*	0 *(0%)*	5 *(0.25%)*	947 **(47.35%)**	1 *(0.05%)*	0 *(0%)*	0 *(0%)*	33 *(1.65%)*	11 *(0.55%)*	303 *(15.15%)*	18 *(0.9%)*	
S8	9 *(0.45%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	1958 **(97.9%)**	0 *(0%)*	0 *(0%)*	0 *(0%)*	1 *(0.05%)*	31 *(1.55%)*	1 *(0.05%)*	
S9	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	2000 **(100%)**	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	
S10	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	2000 **(100%)**	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	
S11	275 *(13.75%)*	4 *(0.2%)*	0 *(0%)*	0 *(0%)*	24 *(1.2%)*	32 *(1.6%)*	16 *(0.8%)*	215 *(10.75%)*	0 *(0%)*	0 *(0%)*	1291 **(64.55%)**	101 *(5.05%)*	0 *(0%)*	42 *(2.1%)*	
S12	148 *(7.4%)*	0 *(0%)*	0 *(0%)*	233 *(11.65%)*	210 *(10.5%)*	195 *(9.75%)*	6 *(0.3%)*	249 *(12.45%)*	60 *(3%)*	0 *(0%)*	488 *(24.4%)*	124 **(6.2%)**	213 *(10.65%)*	74 *(3.7%)*	
S13	52 *(2.6%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	42 *(2.1%)*	14 *(0.7%)*	385 *(19.25%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	42 *(2.1%)*	1381 **(69.05%)**	84 *(4.2%)*	
S14	18 *(0.9%)*	413 *(20.65%)*	0 *(0%)*	0 *(0%)*	7 *(0.35%)*	116 *(5.8%)*	276 *(13.8%)*	9 *(0.45%)*	145 *(7.25%)*	442 *(22.1%)*	427 *(21.35%)*	62 *(3.1%)*	48 *(2.4%)*	37 **(1.85%)**	

Table D.2
Siamese Network: SCADA One-Shot Accuracy (Wrong Connection excluded from Training) Using Different j Votes

No Votes	Overall	New Class (S14)		Normal	
(j)	Accuracy	TPR	FNR	TNR	FPR
1	65.13%	3.75%	0.8%	23.35%	76.65%
5	67.87%	1.85%	0.9%	30%	70%
10	69.32%	1.45%	0.35%	28.4%	71.6%
15	69.37%	1.5%	0.45%	28.65%	71.35%
20	69.8%	1.25%	0.35%	28.2%	71.8%
25	69.8%	0.95%	0.4%	28%	72%
30	70.05%	0.6%	0.3%	28.45%	71.55%

Table D.3
Siamese Network: SCADA One-Shot Confusion Matrix ($j = 5$) (Spoofing excluded from Training)

	Predicted Class														
Correct	Normal (S1)	Blocked measure 1 (S2)	Blocked measure 2 (S3)	DoS (S4)	Humidity (S5)	2 Floating objects (S6)	7 Floating objects (S7)	Person hitting high intensity (S8)	Person hitting med intensity (S9)	Person hitting low intensity (S10)	Plastic bag (S11)	Sensor failure (S12)	Spoofing (S13)	Wrong connection (S14)	Overall
S1	627 **(31.35%)**	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	32 *(1.6%)*	84 *(4.2%)*	903 *(45.15%)*	0 *(0%)*	0 *(0%)*	65 *(3.25%)*	115 *(5.75%)*	167 *(8.35%)*	7 *(0.35%)*	
S2	0 *(0%)*	2000 **(100%)**	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	
S3	0 *(0%)*	0 *(0%)*	2000 **(100%)**	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	
S4	0 *(0%)*	0 *(0%)*	0 *(0%)*	2000 **(100%)**	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	65.87%
S5	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	2000 **(100%)**	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	
S6	11 *(0.55%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	1608 **(80.4%)**	2 *(0.1%)*	1 *(0.05%)*	0 *(0%)*	0 *(0%)*	198 *(9.9%)*	16 *(0.8%)*	164 *(8.2%)*	0 *(0%)*	
S7	157 *(7.85%)*	0 *(0%)*	581 *(29.05%)*	0 *(0%)*	662 *(33.1%)*	0 *(0%)*	359 **(17.95%)**	0 *(0%)*	0 *(0%)*	0 *(0%)*	34 *(1.7%)*	71 *(3.55%)*	131 *(6.55%)*	5 *(0.25%)*	
S8	11 *(0.55%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	1984 **(99.2%)**	0 *(0%)*	0 *(0%)*	4 *(0.2%)*	1 *(0.05%)*	0 *(0%)*	0 *(0%)*	
S9	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	2000 **(100%)**	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	
S10	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	2000 **(100%)**	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	
S11	90 *(4.5%)*	0 *(0%)*	0 *(0%)*	40 *(2%)*	12 *(0.6%)*	283 *(14.15%)*	41 *(2.05%)*	522 *(26.1%)*	0 *(0%)*	0 *(0%)*	447 **(22.35%)**	174 *(8.7%)*	385 *(19.25%)*	6 *(0.3%)*	
S12	404 *(20.2%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	61 *(3.05%)*	40 *(2%)*	68 *(3.4%)*	497 *(24.85%)*	0 *(0%)*	0 *(0%)*	262 *(13.1%)*	421 **(21.05%)**	246 *(12.3%)*	1 *(0.05%)*	
S13	424 *(21.2%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	187 *(9.35%)*	152 *(7.6%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	436 *(21.8%)*	241 *(12.05%)*	558 **(27.9%)**	2 *(0.1%)*	
S14	40 *(2%)*	205 *(10.25%)*	0 *(0%)*	1121 *(56.05%)*	1 *(0.05%)*	38 *(1.9%)*	20 *(1%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	16 *(0.8%)*	11 *(0.55%)*	108 *(5.4%)*	440 **(22%)**	

Table D.4
Siamese Network: SCADA One-Shot Accuracy (Spoofing excluded from Training) Using Different j Votes

No Votes	Overall	New Class (S13)		Normal	
(j)	Accuracy	TPR	FNR	TNR	FPR
1	65.29%	31.25%	15.1%	23.3%	76.7%
5	65.87%	27.9%	21.2%	31.35%	68.65%
10	65.6%	32%	21.2%	29.7%	70.3%
15	65.83%	36.75%	20%	27.5%	72.5%
20	65.63%	39%	19.2%	26.75%	73.25%
25	65.86%	42.55%	18.8%	26.85%	73.15%
30	65.85%	43.95%	18.15%	25.4%	74.6%

Table D.5
Siamese Network: SCADA One-Shot Confusion Matrix ($j = 5$) (Sensor Failure excluded from Training)

	Predicted Class														
Correct	Normal (S1)	Blocked measure 1 (S2)	Blocked measure 2 (S3)	DoS (S4)	Humidity (S5)	2 Floating objects (S6)	7 Floating objects (S7)	Person hitting high intensity (S8)	Person hitting med intensity (S9)	Person hitting low intensity (S10)	Plastic bag (S11)	Sensor failure (S12)	Spoofing (S13)	Wrong connection (S14)	Overall
S1	475 **(23.75%)**	0 *(0%)*	0 *(0%)*	168 *(8.4%)*	4 *(0.2%)*	753 *(37.65%)*	31 *(1.55%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	132 *(6.6%)*	139 *(6.95%)*	35 *(1.75%)*	263 *(13.15%)*	
S2	0 *(0%)*	2000 **(100%)**	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	
S3	0 *(0%)*	0 *(0%)*	2000 **(100%)**	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	
S4	0 *(0%)*	0 *(0%)*	0 *(0%)*	2000 **(100%)**	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	67.44%
S5	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	1996 **(99.8%)**	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	3 *(0.15%)*	0 *(0%)*	1 *(0.05%)*	0 *(0%)*	
S6	340 *(17%)*	0 *(0%)*	0 *(0%)*	33 *(1.65%)*	0 *(0%)*	1513 **(75.65%)**	5 *(0.25%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	59 *(2.95%)*	50 *(2.5%)*	0 *(0%)*	0 *(0%)*	
S7	8 *(0.4%)*	0 *(0%)*	240 *(12%)*	349 *(17.45%)*	264 *(13.2%)*	0 *(0%)*	882 **(44.1%)**	103 *(5.15%)*	0 *(0%)*	0 *(0%)*	83 *(4.15%)*	5 *(0.25%)*	66 *(3.3%)*	0 *(0%)*	
S8	5 *(0.25%)*	0 *(0%)*	0 *(0%)*	39 *(1.95%)*	19 *(0.95%)*	0 *(0%)*	3 *(0.15%)*	1932 **(96.6%)**	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	1 *(0.05%)*	1 *(0.05%)*	
S9	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	2000 **(100%)**	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	
S10	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	2000 **(100%)**	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	
S11	164 *(8.2%)*	1 *(0.05%)*	0 *(0%)*	5 *(0.25%)*	638 *(31.9%)*	428 *(21.4%)*	7 *(0.35%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	300 **(15%)**	41 *(2.05%)*	214 *(10.7%)*	202 *(10.1%)*	
S12	431 *(21.55%)*	0 *(0%)*	0 *(0%)*	117 *(5.85%)*	69 *(3.45%)*	671 *(33.55%)*	27 *(1.35%)*	1 *(0.05%)*	0 *(0%)*	0 *(0%)*	125 *(6.25%)*	281 **(14.05%)**	96 *(4.8%)*	182 *(9.1%)*	
S13	15 *(0.75%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	946 *(47.3%)*	0 *(0%)*	16 *(0.8%)*	229 *(11.45%)*	0 *(0%)*	0 *(0%)*	301 *(15.05%)*	13 *(0.65%)*	475 **(23.75%)**	5 *(0.25%)*	
S14	185 *(9.25%)*	317 *(15.85%)*	0 *(0%)*	67 *(3.35%)*	2 *(0.1%)*	0 *(0%)*	5 *(0.25%)*	210 *(10.5%)*	0 *(0%)*	0 *(0%)*	90 *(4.5%)*	76 *(3.8%)*	18 *(0.9%)*	1030 **(51.5%)**	

Table D.6
Siamese Network: SCADA One-Shot Accuracy (Sensor Failure excluded from Training) Using Different j Votes

No Votes	Overall	New Class (S12)		Normal	
(j)	Accuracy	TPR	FNR	TNR	FPR
1	64.4%	18.5%	18.2%	19.65%	80.35%
5	67.44%	14.05%	21.55%	23.75%	76.25%
10	68.78%	16.15%	14.85%	19.25%	80.75%
15	68.94%	16.45%	12.45%	14.65%	85.35%
20	69.49%	17.4%	10.35%	13.15%	86.85%
25	69.74%	18.7%	8.2%	12.7%	87.3%
30	69.92%	19.95%	7.4%	10.35%	89.65%

Table D.7
Siamese Network: SCADA One-Shot Confusion Matrix ($j = 5$) (Plastic Bag excluded from Training)

	Predicted Class														
Correct	Normal (S1)	Blocked measure 1 (S2)	Blocked measure 2 (S3)	DoS (S4)	Humidity (S5)	2 Floating objects (S6)	7 Floating objects (S7)	Person hitting high intensity (S8)	Person hitting med intensity (S9)	Person hitting low intensity (S10)	Plastic bag (S11)	Sensor failure (S12)	Spoofing (S13)	Wrong connection (S14)	Overall
S1	969 **(48.45%)**	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	408 *(20.4%)*	0 *(0%)*	63 *(3.15%)*	0 *(0%)*	0 *(0%)*	117 *(5.85%)*	166 *(8.3%)*	277 *(13.85%)*	0 *(0%)*	
S2	0 *(0%)*	2000 **(100%)**	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	
S3	0 *(0%)*	0 *(0%)*	2000 **(100%)**	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	
S4	0 *(0%)*	0 *(0%)*	0 *(0%)*	2000 **(100%)**	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	74.4%
S5	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	2000 **(100%)**	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	
S6	259 *(12.95%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	1547 **(77.35%)**	16 *(0.8%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	132 *(6.6%)*	38 *(1.9%)*	8 *(0.4%)*	0 *(0%)*	
S7	1 *(0.05%)*	0 *(0%)*	484 *(24.2%)*	0 *(0%)*	742 *(37.1%)*	58 *(2.9%)*	498 **(24.9%)**	0 *(0%)*	0 *(0%)*	0 *(0%)*	10 *(0.5%)*	53 *(2.65%)*	0 *(0%)*	154 *(7.7%)*	
S8	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	2000 **(100%)**	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	
S9	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	2000 **(100%)**	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	
S10	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	2000 **(100%)**	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	
S11	529 *(26.45%)*	48 *(2.4%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	267 *(13.35%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	338 **(16.9%)**	264 *(13.2%)*	554 *(27.7%)*	0 *(0%)*	
S12	511 *(25.55%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	112 *(5.6%)*	301 *(15.05%)*	138 *(6.9%)*	423 *(21.15%)*	0 *(0%)*	0 *(0%)*	153 *(7.65%)*	250 **(12.5%)**	107 *(5.35%)*	5 *(0.25%)*	
S13	32 *(1.6%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	3 *(0.15%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	117 *(5.85%)*	11 *(0.55%)*	1837 **(91.85%)**	0 *(0%)*	
S14	9 *(0.45%)*	310 *(15.5%)*	0 *(0%)*	0 *(0%)*	212 *(10.6%)*	0 *(0%)*	52 *(2.6%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	1 *(0.05%)*	2 *(0.1%)*	21 *(1.05%)*	1393 **(69.65%)**	

Table D.8
Siamese Network: SCADA One-Shot Accuracy (Plastic Bag excluded from Training) Using Different j Votes

No Votes	Overall	New Class (S11)		Normal	
(j)	Accuracy	TPR	FNR	TNR	FPR
1	71.05%	19.05%	19.2%	35.15%	64.85%
5	74.4%	16.9%	26.45%	48.45%	51.55%
10	74.83%	12.15%	27.4%	48.15%	51.85%
15	75.15%	12.3%	27.15%	46.7%	53.3%
20	75.24%	11.45%	26.1%	47.5%	52.5%
25	75.23%	10.8%	25.3%	46.85%	53.15%
30	75.23%	10.7%	25.45%	47.45%	52.55%

Table D.9
Siamese Network: SCADA One-Shot Confusion Matrix ($j = 5$) (Person Hitting Low Intensity excluded from Training)

	Predicted Class														
Correct	Normal (S1)	Blocked measure 1 (S2)	Blocked measure 2 (S3)	DoS (S4)	Humidity (S5)	2 Floating objects (S6)	7 Floating objects (S7)	Person hitting high intensity (S8)	Person hitting med intensity (S9)	Person hitting low intensity (S10)	Plastic bag (S11)	Sensor failure (S12)	Spoofing (S13)	Wrong connection (S14)	Overall
S1	455 **(22.75%)**	0 *(0%)*	0 *(0%)*	38 *(1.9%)*	24 *(1.2%)*	335 *(16.75%)*	11 *(0.55%)*	421 *(21.05%)*	278 *(13.9%)*	0 *(0%)*	130 *(6.5%)*	73 *(3.65%)*	21 *(1.05%)*	214 *(10.7%)*	
S2	0 *(0%)*	2000 **(100%)**	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	
S3	0 *(0%)*	0 *(0%)*	2000 **(100%)**	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	
S4	0 *(0%)*	0 *(0%)*	0 *(0%)*	2000 **(100%)**	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	61.79%
S5	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	1978 **(98.9%)**	0 *(0%)*	15 *(0.75%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	3 *(0.15%)*	0 *(0%)*	4 *(0.2%)*	0 *(0%)*	
S6	181 *(9.05%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	634 **(31.7%)**	0 *(0%)*	790 *(39.5%)*	334 *(16.7%)*	0 *(0%)*	46 *(2.3%)*	15 *(0.75%)*	0 *(0%)*	0 *(0%)*	
S7	11 *(0.55%)*	0 *(0%)*	431 *(21.55%)*	0 *(0%)*	601 *(30.05%)*	0 *(0%)*	326 **(16.3%)**	0 *(0%)*	226 *(11.3%)*	194 *(9.7%)*	66 *(3.3%)*	12 *(0.6%)*	131 *(6.55%)*	2 *(0.1%)*	
S8	17 *(0.85%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	70 *(3.5%)*	0 *(0%)*	1911 **(95.55%)**	0 *(0%)*	0 *(0%)*	1 *(0.05%)*	1 *(0.05%)*	0 *(0%)*	0 *(0%)*	
S9	1 *(0.05%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	1 *(0.05%)*	0 *(0%)*	0 *(0%)*	1998 **(99.9%)**	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	
S10	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	51 *(2.55%)*	0 *(0%)*	0 *(0%)*	1947 **(97.35%)**	0 *(0%)*	0 *(0%)*	0 *(0%)*	2 *(0.1%)*	
S11	186 *(9.3%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	784 *(39.2%)*	204 *(10.2%)*	75 *(3.75%)*	298 *(14.9%)*	52 *(2.6%)*	0 *(0%)*	168 **(8.4%)**	38 *(1.9%)*	116 *(5.8%)*	79 *(3.95%)*	
S12	195 *(9.75%)*	0 *(0%)*	0 *(0%)*	816 *(40.8%)*	145 *(7.25%)*	171 *(8.55%)*	22 *(1.1%)*	215 *(10.75%)*	13 *(0.65%)*	0 *(0%)*	64 *(3.2%)*	187 **(9.35%)**	30 *(1.5%)*	142 *(7.1%)*	
S13	1 *(0.05%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	889 *(44.45%)*	0 *(0%)*	140 *(7%)*	0 *(0%)*	0 *(0%)*	279 *(13.95%)*	169 *(8.45%)*	14 *(0.7%)*	507 **(25.35%)**	1 *(0.05%)*	
S14	147 *(7.35%)*	334 *(16.7%)*	0 *(0%)*	0 *(0%)*	5 *(0.25%)*	0 *(0%)*	6 *(0.3%)*	0 *(0%)*	22 *(1.1%)*	204 *(10.2%)*	38 *(1.9%)*	47 *(2.35%)*	7 *(0.35%)*	1190 **(59.5%)**	

Table D.10
Siamese Network: SCADA One-Shot Accuracy (Person Hitting Low Intensity excluded from Training) Using Different j Votes

No Votes	Overall	New Class (S10)		Normal	
(j)	Accuracy	TPR	FNR	TNR	FPR
1	58.55%	85.9%	0%	15.5%	84.5%
5	61.79%	97.35%	0%	22.75%	77.25%
10	61.79%	99.15%	0%	20.1%	79.9%
15	61.77%	99.8%	0%	19.1%	80.9%
20	61.5%	99.95%	0%	18.05%	81.95%
25	61.49%	99.95%	0%	18.3%	81.7%
30	61.2%	100%	0%	17.65%	82.35%

Table D.11
Siamese Network: SCADA One-Shot Confusion Matrix ($j = 5$) (Person Hitting Medium Intensity excluded from Training)

	Predicted Class														
Correct	Normal (S1)	Blocked measure 1 (S2)	Blocked measure 2 (S3)	DoS (S4)	Humidity (S5)	2 Floating objects (S6)	7 Floating objects (S7)	Person hitting high intensity (S8)	Person hitting med intensity (S9)	Person hitting low intensity (S10)	Plastic bag (S11)	Sensor failure (S12)	Spoofing (S13)	Wrong connection (S14)	Overall
S1	255 **(12.75%)**	0 *(0%)*	0 *(0%)*	87 *(4.35%)*	220 *(11%)*	317 *(15.85%)*	25 *(1.25%)*	218 *(10.9%)*	13 *(0.65%)*	0 *(0%)*	155 *(7.75%)*	26 *(1.3%)*	216 *(10.8%)*	468 *(23.4%)*	
S2	0 *(0%)*	1997 **(99.85%)**	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	3 *(0.15%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	
S3	0 *(0%)*	0 *(0%)*	2000 **(100%)**	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	
S4	0 *(0%)*	0 *(0%)*	0 *(0%)*	2000 **(100%)**	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	58.24%
S5	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	1999 **(99.95%)**	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	1 *(0.05%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	
S6	206 *(10.3%)*	0 *(0%)*	0 *(0%)*	4 *(0.2%)*	107 *(5.35%)*	898 **(44.9%)**	0 *(0%)*	106 *(5.3%)*	0 *(0%)*	0 *(0%)*	71 *(3.55%)*	41 *(2.05%)*	566 *(28.3%)*	1 *(0.05%)*	
S7	37 *(1.85%)*	395 *(19.75%)*	458 *(22.9%)*	0 *(0%)*	94 *(4.7%)*	0 *(0%)*	279 **(13.95%)**	0 *(0%)*	285 *(14.25%)*	100 *(5%)*	260 *(13%)*	31 *(1.55%)*	0 *(0%)*	61 *(3.05%)*	
S8	9 *(0.45%)*	0 *(0%)*	0 *(0%)*	669 *(33.45%)*	0 *(0%)*	6 *(0.3%)*	0 *(0%)*	1313 **(65.65%)**	0 *(0%)*	0 *(0%)*	0 *(0%)*	3 *(0.15%)*	0 *(0%)*	0 *(0%)*	
S9	2 *(0.1%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	4 *(0.2%)*	0 *(0%)*	1758 **(87.9%)**	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	236 *(11.8%)*	
S10	0 *(0%)*	895 *(44.75%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	1105 **(55.25%)**	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	
S11	102 *(5.1%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	531 *(26.55%)*	203 *(10.15%)*	138 *(6.9%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	726 **(36.3%)**	46 *(2.3%)*	238 *(11.9%)*	16 *(0.8%)*	
S12	98 *(4.9%)*	0 *(0%)*	0 *(0%)*	669 *(33.45%)*	65 *(3.25%)*	189 *(9.45%)*	68 *(3.4%)*	135 *(6.75%)*	0 *(0%)*	0 *(0%)*	320 *(16%)*	340 **(17%)**	106 *(5.3%)*	10 *(0.5%)*	
S13	126 *(6.3%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	357 *(17.85%)*	660 *(33%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	76 *(3.8%)*	14 *(0.7%)*	767 **(38.35%)**	0 *(0%)*	
S14	251 *(12.55%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	9 *(0.45%)*	0 *(0%)*	25 *(1.25%)*	0 *(0%)*	770 *(38.5%)*	0 *(0%)*	48 *(2.4%)*	20 *(1%)*	8 *(0.4%)*	869 **(43.45%)**	

Table D.12
Siamese Network: SCADA One-Shot Accuracy (Person Hitting Medium Intensity excluded from Training) Using Different j Votes

No Votes	Overall	New Class (S9)		Normal	
(j)	Accuracy	TPR	FNR	TNR	FPR
1	56.04%	73.8%	0.3%	13.4%	86.6%
5	58.24%	87.9%	0.1%	12.75%	87.25%
10	59.55%	96.5%	0%	7.3%	92.7%
15	60.1%	97.1%	0%	4.5%	95.5%
20	60.27%	98.7%	0%	2.5%	97.5%
25	60.63%	98.8%	0%	1.5%	98.5%
30	60.58%	99.25%	0%	1.3%	98.7%

Table D.13
Siamese Network: SCADA One-Shot Confusion Matrix ($j = 5$) (7 Floating Objects excluded from Training)

	Predicted Class														
Correct	Normal (S1)	Blocked measure 1 (S2)	Blocked measure 2 (S3)	DoS (S4)	Humidity (S5)	2 Floating objects (S6)	7 Floating objects (S7)	Person hitting high intensity (S8)	Person hitting med intensity (S9)	Person hitting low intensity (S10)	Plastic bag (S11)	Sensor failure (S12)	Spoofing (S13)	Wrong connection (S14)	Overall
S1	1222 **(61.1%)**	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	74 *(3.7%)*	0 *(0%)*	0 *(0%)*	392 *(19.6%)*	223 *(11.15%)*	89 *(4.45%)*	0 *(0%)*	
S2	0 *(0%)*	1999 **(99.95%)**	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	1 *(0.05%)*	0 *(0%)*	0 *(0%)*	
S3	0 *(0%)*	0 *(0%)*	2000 **(100%)**	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	
S4	0 *(0%)*	0 *(0%)*	0 *(0%)*	2000 **(100%)**	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	81.49%
S5	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	2000 **(100%)**	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	
S6	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	1996 **(99.8%)**	4 *(0.2%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	
S7	16 *(0.8%)*	0 *(0%)*	650 *(32.5%)*	0 *(0%)*	1 *(0.05%)*	101 *(5.05%)*	1089 **(54.45%)**	23 *(1.15%)*	0 *(0%)*	0 *(0%)*	51 *(2.55%)*	1 *(0.05%)*	66 *(3.3%)*	2 *(0.1%)*	
S8	30 *(1.5%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	36 *(1.8%)*	6 *(0.3%)*	1920 **(96%)**	0 *(0%)*	0 *(0%)*	7 *(0.35%)*	1 *(0.05%)*	0 *(0%)*	0 *(0%)*	
S9	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	2000 **(100%)**	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	
S10	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	2000 **(100%)**	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	
S11	499 *(24.95%)*	53 *(2.65%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	4 *(0.2%)*	46 *(2.3%)*	32 *(1.6%)*	0 *(0%)*	0 *(0%)*	929 **(46.45%)**	101 *(5.05%)*	336 *(16.8%)*	0 *(0%)*	
S12	645 *(32.25%)*	469 *(23.45%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	48 *(2.4%)*	0 *(0%)*	0 *(0%)*	247 *(12.35%)*	324 **(16.2%)**	267 *(13.35%)*	0 *(0%)*	
S13	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	3 *(0.15%)*	1 *(0.05%)*	1996 **(99.8%)**	0 *(0%)*	
S14	0 *(0%)*	330 *(16.5%)*	0 *(0%)*	0 *(0%)*	39 *(1.95%)*	0 *(0%)*	249 *(12.45%)*	37 *(1.85%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	3 *(0.15%)*	0 *(0%)*	1342 **(67.1%)**	

Table D.14
Siamese Network: SCADA One-Shot Accuracy (7 Floating Objects excluded from Training) Using Different j Votes

No Votes	Overall	New Class (S7)		Normal	
(j)	Accuracy	TPR	FNR	TNR	FPR
1	78.27%	48.25%	0.95%	43.8%	56.2%
5	81.49%	54.45%	0.8%	61.1%	38.9%
10	82.29%	54.45%	0.65%	66.95%	33.05%
15	82.87%	54.35%	0.7%	70.35%	29.65%
20	83.13%	54.45%	0.65%	72.55%	27.45%
25	83.28%	54.5%	0.75%	74.3%	25.7%
30	83.38%	54.4%	0.75%	75.05%	24.95%

Table D.15
Siamese Network: SCADA One-Shot Confusion Matrix ($j = 5$) (2 Floating Objects excluded from Training)

	Predicted Class														
Correct	Normal (S1)	Blocked measure 1 (S2)	Blocked measure 2 (S3)	DoS (S4)	Humidity (S5)	2 Floating objects (S6)	7 Floating objects (S7)	Person hitting high intensity (S8)	Person hitting med intensity (S9)	Person hitting low intensity (S10)	Plastic bag (S11)	Sensor failure (S12)	Spoofing (S13)	Wrong connection (S14)	Overall
S1	1142 **(57.1%)**	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	45 *(2.25%)*	18 *(0.9%)*	368 *(18.4%)*	0 *(0%)*	0 *(0%)*	339 *(16.95%)*	86 *(4.3%)*	2 *(0.1%)*	0 *(0%)*	
S2	0 *(0%)*	2000 **(100%)**	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	
S3	0 *(0%)*	0 *(0%)*	2000 **(100%)**	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	
S4	0 *(0%)*	0 *(0%)*	0 *(0%)*	2000 **(100%)**	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	72.81%
S5	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	2000 **(100%)**	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	
S6	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	401 **(20.05%)**	96 *(4.8%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	1503 *(75.15%)*	0 *(0%)*	
S7	12 *(0.6%)*	36 *(1.8%)*	469 *(23.45%)*	0 *(0%)*	0 *(0%)*	53 *(2.65%)*	528 **(26.4%)**	0 *(0%)*	0 *(0%)*	0 *(0%)*	76 *(3.8%)*	34 *(1.7%)*	785 *(39.25%)*	7 *(0.35%)*	
S8	76 *(3.8%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	34 *(1.7%)*	0 *(0%)*	1860 **(93%)**	0 *(0%)*	0 *(0%)*	2 *(0.1%)*	28 *(1.4%)*	0 *(0%)*	0 *(0%)*	
S9	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	2000 **(100%)**	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	
S10	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	2000 **(100%)**	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	
S11	538 *(26.9%)*	52 *(2.6%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	72 *(3.6%)*	74 *(3.7%)*	53 *(2.65%)*	0 *(0%)*	0 *(0%)*	975 **(48.75%)**	80 *(4%)*	153 *(7.65%)*	3 *(0.15%)*	
S12	346 *(17.3%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	418 *(20.9%)*	18 *(0.9%)*	65 *(3.25%)*	628 *(31.4%)*	0 *(0%)*	0 *(0%)*	246 *(12.3%)*	103 **(5.15%)**	173 *(8.65%)*	3 *(0.15%)*	
S13	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	46 *(2.3%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	9 *(0.45%)*	5 *(0.25%)*	1937 **(96.85%)**	3 *(0.15%)*	
S14	0 *(0%)*	311 *(15.55%)*	0 *(0%)*	0 *(0%)*	17 *(0.85%)*	0 *(0%)*	8 *(0.4%)*	23 *(1.15%)*	0 *(0%)*	0 *(0%)*	6 *(0.3%)*	8 *(0.4%)*	186 *(9.3%)*	1441 **(72.05%)**	

Table D.16
Siamese Network: SCADA One-Shot Accuracy (2 Floating Objects excluded from Training) Using Different j Votes

No Votes	Overall	New Class (S6)		Normal	
(j)	Accuracy	TPR	FNR	TNR	FPR
1	69.9%	21.05%	0%	38.5%	61.5%
5	72.81%	20.05%	0%	57.1%	42.9%
10	73.1%	17.6%	0%	58.85%	41.15%
15	73%	14.8%	0%	60.35%	39.65%
20	72.89%	13.1%	0%	61.6%	38.4%
25	72.73%	12.4%	0%	61.55%	38.45%
30	72.59%	11.25%	0%	61.75%	38.25%

Table D.17
Siamese Network: SCADA One-Shot Confusion Matrix ($j = 5$) (Humidity excluded from Training)

	Predicted Class														
Correct	Normal (S1)	Blocked measure 1 (S2)	Blocked measure 2 (S3)	DoS (S4)	Humidity (S5)	2 Floating objects (S6)	7 Floating objects (S7)	Person hitting high intensity (S8)	Person hitting med intensity (S9)	Person hitting low intensity (S10)	Plastic bag (S11)	Sensor failure (S12)	Spoofing (S13)	Wrong connection (S14)	Overall
S1	198 **(9.9%)**	0 *(0%)*	0 *(0%)*	240 *(12%)*	206 *(10.3%)*	239 *(11.95%)*	20 *(1%)*	196 *(9.8%)*	0 *(0%)*	0 *(0%)*	149 *(7.45%)*	33 *(1.65%)*	221 *(11.05%)*	498 *(24.9%)*	
S2	0 *(0%)*	1999 **(99.95%)**	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	1 *(0.05%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	
S3	0 *(0%)*	0 *(0%)*	2000 **(100%)**	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	
S4	0 *(0%)*	0 *(0%)*	0 *(0%)*	2000 **(100%)**	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	60.45%
S5	13 *(0.65%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	1937 **(96.85%)**	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	2 *(0.1%)*	2 *(0.1%)*	46 *(2.3%)*	0 *(0%)*	
S6	170 *(8.5%)*	0 *(0%)*	0 *(0%)*	470 *(23.5%)*	77 *(3.85%)*	658 **(32.9%)**	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	59 *(2.95%)*	24 *(1.2%)*	542 *(27.1%)*	0 *(0%)*	
S7	28 *(1.4%)*	285 *(14.25%)*	518 *(25.9%)*	0 *(0%)*	120 *(6%)*	0 *(0%)*	243 **(12.15%)**	0 *(0%)*	423 *(21.15%)*	0 *(0%)*	288 *(14.4%)*	30 *(1.5%)*	0 *(0%)*	65 *(3.25%)*	
S8	8 *(0.4%)*	0 *(0%)*	0 *(0%)*	17 *(0.85%)*	0 *(0%)*	4 *(0.2%)*	0 *(0%)*	1959 **(97.95%)**	0 *(0%)*	0 *(0%)*	0 *(0%)*	12 *(0.6%)*	0 *(0%)*	0 *(0%)*	
S9	0 *(0%)*	883 *(44.15%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	45 *(2.25%)*	0 *(0%)*	1072 **(53.6%)**	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	
S10	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	2000 **(100%)**	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	
S11	110 *(5.5%)*	0 *(0%)*	0 *(0%)*	1 *(0.05%)*	547 *(27.35%)*	208 *(10.4%)*	148 *(7.4%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	598 **(29.9%)**	82 *(4.1%)*	244 *(12.2%)*	62 *(3.1%)*	
S12	92 *(4.6%)*	0 *(0%)*	0 *(0%)*	127 *(6.35%)*	74 *(3.7%)*	144 *(7.2%)*	64 *(3.2%)*	783 *(39.15%)*	0 *(0%)*	0 *(0%)*	287 *(14.35%)*	298 **(14.9%)**	111 *(5.55%)*	20 *(1%)*	
S13	147 *(7.35%)*	0 *(0%)*	0 *(0%)*	4 *(0.2%)*	408 *(20.4%)*	607 *(30.35%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	82 *(4.1%)*	32 *(1.6%)*	720 **(36%)**	0 *(0%)*	
S14	254 *(12.7%)*	16 *(0.8%)*	0 *(0%)*	0 *(0%)*	24 *(1.2%)*	1 *(0.05%)*	137 *(6.85%)*	0 *(0%)*	188 *(9.4%)*	0 *(0%)*	99 *(4.95%)*	30 *(1.5%)*	6 *(0.3%)*	1245 **(62.25%)**	

Table D.18
Siamese Network: SCADA One-Shot Accuracy (Humidity excluded from Training) Using Different j Votes

No Votes	Overall	New Class (S5)		Normal	
(j)	Accuracy	TPR	FNR	TNR	FPR
1	58.44%	80.9%	2.85%	12.9%	87.1%
5	60.45%	96.85%	0.65%	9.9%	90.1%
10	62.07%	99.35%	0.05%	4.3%	95.7%
15	62.65%	99.95%	0%	2.7%	97.3%
20	62.91%	100%	0%	1.6%	98.4%
25	63.16%	100%	0%	0.5%	99.5%
30	63.17%	100%	0%	0.45%	99.55%

Table D.19
Siamese Network: SCADA One-Shot Confusion Matrix ($j = 5$) (Blocked Measure 2 excluded from Training)

	Predicted Class														
Correct	Normal (S1)	Blocked measure 1 (S2)	Blocked measure 2 (S3)	DoS (S4)	Humidity (S5)	2 Floating objects (S6)	7 Floating objects (S7)	Person hitting high intensity (S8)	Person hitting med intensity (S9)	Person hitting low intensity (S10)	Plastic bag (S11)	Sensor failure (S12)	Spoofing (S13)	Wrong connection (S14)	Overall
S1	1136 **(56.8%)**	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	482 *(24.1%)*	0 *(0%)*	0 *(0%)*	175 *(8.75%)*	207 *(10.35%)*	0 *(0%)*	0 *(0%)*	
S2	0 *(0%)*	2000 **(100%)**	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	
S3	0 *(0%)*	0 *(0%)*	2000 **(100%)**	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	
S4	0 *(0%)*	0 *(0%)*	0 *(0%)*	2000 **(100%)**	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	78.96%
S5	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	2000 **(100%)**	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	
S6	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	1618 **(80.9%)**	8 *(0.4%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	374 *(18.7%)*	0 *(0%)*	
S7	6 *(0.3%)*	8 *(0.4%)*	17 *(0.85%)*	0 *(0%)*	0 *(0%)*	106 *(5.3%)*	1390 **(69.5%)**	9 *(0.45%)*	0 *(0%)*	0 *(0%)*	37 *(1.85%)*	5 *(0.25%)*	421 *(21.05%)*	1 *(0.05%)*	
S8	57 *(2.85%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	11 *(0.55%)*	1903 **(95.15%)**	0 *(0%)*	0 *(0%)*	17 *(0.85%)*	12 *(0.6%)*	0 *(0%)*	0 *(0%)*	
S9	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	2000 **(100%)**	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	
S10	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	2000 **(100%)**	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	
S11	546 *(27.3%)*	52 *(2.6%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	1 *(0.05%)*	96 *(4.8%)*	334 *(16.7%)*	0 *(0%)*	0 *(0%)*	757 **(37.85%)**	88 *(4.4%)*	121 *(6.05%)*	5 *(0.25%)*	
S12	474 *(23.7%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	414 *(20.7%)*	0 *(0%)*	127 *(6.35%)*	456 *(22.8%)*	0 *(0%)*	0 *(0%)*	234 *(11.7%)*	223 **(11.15%)**	68 *(3.4%)*	4 *(0.2%)*	
S13	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	239 *(11.95%)*	79 *(3.95%)*	0 *(0%)*	0 *(0%)*	0 *(0%)*	5 *(0.25%)*	2 *(0.1%)*	1674 **(83.7%)**	1 *(0.05%)*	
S14	1 *(0.05%)*	311 *(15.55%)*	0 *(0%)*	0 *(0%)*	36 *(1.8%)*	0 *(0%)*	51 *(2.55%)*	13 *(0.65%)*	0 *(0%)*	0 *(0%)*	22 *(1.1%)*	12 *(0.6%)*	145 *(7.25%)*	1409 **(70.45%)**	

Table D.20
Siamese Network: SCADA One-Shot Accuracy (Blocked Measure 2 excluded from Training) Using Different j Votes

No Votes	Overall	New Class (S3)		Normal	
(j)	Accuracy	TPR	FNR	TNR	FPR
1	75.99%	100%	0%	43.4%	56.6%
5	78.96%	100%	0%	56.8%	43.2%
10	80.15%	100%	0%	61.3%	38.7%
15	80.7%	100%	0%	62.65%	37.35%
20	80.94%	100%	0%	64.6%	35.4%
25	81.1%	100%	0%	65.2%	34.8%
30	81.23%	100%	0%	66.3%	33.7%

D.2 CICIDS2017 Dataset

Table D.21
Siamese Network: CICIDS2017 One-Shot Confusion Matrix ($j = 5$) (DoS (Slowloris) excluded from training)

	Predicted Class					
Correct	Normal	DoS (Hulk)	DoS (Slowloris)	FTP	SSH	Overall
Normal	5307 **(88.45%)**	6 *(0.1%)*	459 *(7.65%)*	64 *(1.07%)*	164 *(2.73%)*	81.07%
DoS (Hulk)	37 *(0.62%)*	5794 **(96.57%)**	65 *(1.08%)*	53 *(0.88%)*	51 *(0.85%)*	
DoS (Slowloris)	574 *(9.57%)*	26 *(0.43%)*	4024 **(67.07%)**	582 *(9.7%)*	794 *(13.23%)*	
FTP	482 *(8.03%)*	1 *(0.02%)*	598 *(9.97%)*	4639 **(77.32%)**	280 *(4.67%)*	
SSH	446 *(7.43%)*	0 *(0%)*	817 *(13.62%)*	181 *(3.02%)*	4556 **(75.93%)**	

Table D.22
Siamese Network: CICIDS2017 One-Shot Accuracy (DoS (Slowloris) excluded from Training) Using Different j Votes

No Votes	Overall	New Class (DoS (Slowloris))		Normal	
(j)	Accuracy	TPR	FNR	TNR	FPR
1	70.89%	50.97%	11.50%	72.65%	27.35%
5	81.07%	67.07%	9.57%	88.45%	11.55%
10	82.67%	71.38%	7.38%	89.48%	10.52%
15	82.85%	72.20%	7.18%	89.37%	10.63%
20	83.01%	72.77%	6.85%	89.67%	10.33%
25	82.98%	72.93%	6.58%	89.65%	10.35%
30	82.94%	72.82%	6.68%	89.70%	10.30%

Table D.23
Siamese Network: CICIDS2017 One-Shot Confusion Matrix ($j = 5$) (DoS (Hulk) excluded from training)

	Predicted Class					
Correct	Normal	DoS (Hulk)	DoS (Slowloris)	FTP	SSH	Overall
Normal	4314 **(71.9%)**	1095 *(18.25%)*	174 *(2.9%)*	113 *(1.88%)*	304 *(5.07%)*	80.81%
DoS (Hulk)	78 *(1.3%)*	5708 **(95.13%)**	60 *(1%)*	58 *(0.97%)*	96 *(1.6%)*	
DoS (Slowloris)	451 *(7.52%)*	51 *(0.85%)*	4767 **(79.45%)**	111 *(1.85%)*	620 *(10.33%)*	
FTP	624 *(10.4%)*	171 *(2.85%)*	138 *(2.3%)*	4521 **(75.35%)**	546 *(9.1%)*	
SSH	597 *(9.95%)*	26 *(0.43%)*	245 *(4.08%)*	198 *(3.3%)*	4934 **(82.23%)**	

Table D.24
Siamese Network: CICIDS2017 One-Shot Accuracy (DoS (Hulk) excluded from Training) Using Different j Votes

No Votes	Overall	New Class (DoS (Hulk))		Normal	
(j)	Accuracy	TPR	FNR	TNR	FPR
1	72.28%	91.07%	4.90%	58.05%	41.95%
5	80.81%	95.13%	1.30%	71.90%	28.10%
10	82.59%	95.22%	1.22%	75.58%	24.42%
15	82.54%	95.23%	1.20%	74.67%	25.33%
20	82.86%	95.20%	1.20%	76.02%	23.98%
25	82.76%	95.20%	1.15%	75.50%	24.50%
30	82.93%	95.18%	1.22%	76.15%	23.85%

Table D.25
Siamese Network: KDD Cup'99 One-Shot Confusion Matrix ($j = 5$) (U2R excluded from Training)

	Predicted Class					
Correct	Normal	DoS	Probe	R2L	U2R	Overall
Normal	4146 *(69.1%)*	5 *(0.08%)*	440 *(7.33%)*	796 *(13.27%)*	613 *(10.22%)*	75.72%
DoS	7 *(0.12%)*	5921 *(98.68%)*	59 *(0.98%)*	6 *(0.1%)*	7 *(0.12%)*	
Probe	53 *(0.88%)*	384 *(6.4%)*	5449 *(90.82%)*	59 *(0.98%)*	55 *(0.92%)*	
R2L	35 *(0.58%)*	0 *(0%)*	13 *(0.22%)*	5849 *(97.48%)*	103 *(1.72%)*	
U2R	958 *(15.97%)*	1 *(0.02%)*	669 *(11.15%)*	3022 *(50.37%)*	1350 *(22.5%)*	

D.3 KDD Cup'99 Dataset

Table D.26
Siamese Network: KDD Cup'99 One-Shot Accuracy (U2R excluded from Training) Using Different j Votes

No Votes	Overall	New Class (U2R)		Normal	
(j)	Accuracy	TPR	FNR	TNR	FPR
1	70.69%	21.40%	17.28%	59.27%	40.73%
5	75.72%	22.50%	15.97%	69.10%	30.90%
10	76.26%	21.82%	17.17%	72.18%	27.82%
15	76.33%	21.83%	17.15%	72.52%	27.48%
20	76.31%	21.48%	17.52%	72.72%	27.28%
25	76.34%	21.45%	17.55%	72.77%	27.23%
30	76.33%	21.27%	17.73%	72.90%	27.10%

Table D.27
Siamese Network: KDD Cup'99 One-Shot Confusion Matrix ($j = 5$) (R2L excluded from Training)

	Predicted Class					
Correct	Normal	DoS	Probe	R2L	U2R	Overall
Normal	4288 **(71.47%)**	1 *(0.02%)*	400 *(6.67%)*	730 *(12.17%)*	581 *(9.68%)*	74.2%
DoS	10 *(0.17%)*	5909 **(98.48%)**	72 *(1.2%)*	9 *(0.15%)*	0 *(0%)*	
Probe	90 *(1.5%)*	160 *(2.67%)*	5338 **(88.97%)**	165 *(2.75%)*	247 *(4.12%)*	
R2L	1702 *(28.37%)*	2 *(0.03%)*	1344 *(22.4%)*	2148 **(35.8%)**	804 *(13.4%)*	
U2R	527 *(8.78%)*	1 *(0.02%)*	682 *(11.37%)*	213 *(3.55%)*	4577 **(76.28%)**	

Table D.28
Siamese Network: KDD Cup'99 One-Shot Accuracy (R2L excluded from Training) Using Different j Votes

No Votes	Overall	New Class (R2L)		Normal	
(j)	Accuracy	TPR	FNR	TNR	FPR
1	67.75%	38.48%	25.95%	59.65%	40.35%
5	74.2%	35.80%	28.37%	71.47%	28.53%
10	77.27%	42.22%	23.85%	74.38%	25.62%
15	78.34%	46.65%	22.05%	74.50%	25.50%
20	78.94%	49.18%	21.45%	74.62%	25.38%
25	79.44%	51.32%	20.72%	74.65%	25.35%
30	79.87%	53.35%	20.65%	74.55%	25.45%

Table D.29
Siamese Network: KDD Cup'99 One-Shot Confusion Matrix ($j = 5$) (Probe excluded from Training)

	Predicted Class					
Correct	Normal	DoS	Probe	R2L	U2R	Overall
Normal	4515 **(75.25%)**	16 *(0.27%)*	383 *(6.38%)*	1016 *(16.93%)*	70 *(1.17%)*	72.23%
DoS	18 *(0.3%)*	5896 **(98.27%)**	81 *(1.35%)*	4 *(0.07%)*	1 *(0.02%)*	
Probe	719 *(11.98%)*	3707 *(61.78%)*	612 **(10.2%)**	941 *(15.68%)*	21 *(0.35%)*	
R2L	26 *(0.43%)*	0 *(0%)*	16 *(0.27%)*	5946 **(99.1%)**	12 *(0.2%)*	
U2R	55 *(0.92%)*	37 *(0.62%)*	264 *(4.4%)*	943 *(15.72%)*	4701 **(78.35%)**	

Table D.30
Siamese Network: KDD Cup'99 One-Shot Accuracy (Probe excluded from Training) Using Different j Votes

No Votes	Overall	New Class (Probe)		Normal	
(j)	Accuracy	TPR	FNR	TNR	FPR
1	66.72%	15.72%	11.77%	65.72%	34.28%
5	72.23%	10.20%	11.98%	75.25%	24.75%
10	72.59%	5.90%	13.30%	78.65%	21.35%
15	72.35%	4.82%	13.08%	78.57%	21.43%
20	72.26%	3.58%	13.50%	79.20%	20.80%
25	72.17%	3.05%	13.55%	79.23%	20.77%
30	72.07%	2.17%	13.98%	79.62%	20.38%

Table D.31
Siamese Network: NSL-KDD Cup'99 One-Shot Confusion Matrix ($j = 5$) (U2R excluded from Training)

	Predicted Class					
Correct	Normal	DoS	Probe	R2L	U2R	Overall
Normal	4530 **(75.5%)**	127 *(2.12%)*	76 *(1.27%)*	237 *(3.95%)*	1030 *(17.17%)*	77.04%
DoS	120 *(2%)*	5771 **(96.18%)**	49 *(0.82%)*	16 *(0.27%)*	44 *(0.73%)*	
Probe	43 *(0.72%)*	304 *(5.07%)*	5574 **(92.9%)**	69 *(1.15%)*	10 *(0.17%)*	
R2L	403 *(6.72%)*	1 *(0.02%)*	27 *(0.45%)*	5238 **(87.3%)**	331 *(5.52%)*	
U2R	2191 *(36.52%)*	0 *(0%)*	221 *(3.68%)*	1589 *(26.48%)*	1999 **(33.32%)**	

D.4 NSL-KDD Dataset

Table D.32
Siamese Network: NSL-KDD One-Shot Accuracy (U2R excluded from Training) Using Different j Votes

No Votes	Overall	New Class (U2R)		Normal	
(j)	Accuracy	TPR	FNR	TNR	FPR
1	72.42%	34.37%	35.55%	66.58%	33.42%
5	77.04%	33.32%	36.52%	75.50%	24.50%
10	77.08%	30.42%	36.95%	77.85%	22.15%
15	77.19%	30.20%	36.70%	78.22%	21.78%
20	77.12%	29.37%	36.67%	78.52%	21.48%
25	77.14%	28.85%	36.72%	78.87%	21.13%
30	77.12%	28.30%	37.10%	79.25%	20.75%

Table D.33
Siamese Network: NSL-KDD Cup'99 One-Shot Confusion Matrix ($j = 5$) (R2L excluded from Training)

	Predicted Class					
Correct	Normal	DoS	Probe	R2L	U2R	Overall
Normal	5199 **(86.65%)**	24 *(0.4%)*	148 *(2.47%)*	530 *(8.83%)*	99 *(1.65%)*	80.16%
DoS	15 *(0.25%)*	5799 **(96.65%)**	36 *(0.6%)*	26 *(0.43%)*	124 *(2.07%)*	
Probe	90 *(1.5%)*	242 *(4.03%)*	5416 **(90.27%)**	236 *(3.93%)*	16 *(0.27%)*	
R2L	2526 *(42.1%)*	1 *(0.02%)*	142 *(2.37%)*	2759 **(45.98%)**	572 *(9.53%)*	
U2R	852 *(14.2%)*	3 *(0.05%)*	0 *(0%)*	270 *(4.5%)*	4875 **(81.25%)**	

Table D.34
Siamese Network: NSL-KDD One-Shot Accuracy (R2L excluded from Training) Using Different j Votes

No Votes	Overall	New Class (R2L)		Normal	
(j)	Accuracy	TPR	FNR	TNR	FPR
1	74.5%	46.05%	38.13%	74.73%	25.27%
5	80.16%	45.98%	42.10%	86.65%	13.35%
10	80.79%	46.82%	41.58%	88.07%	11.93%
15	81.09%	49.02%	39.88%	87.72%	12.28%
20	81%	48.62%	40.38%	87.90%	12.10%
25	80.95%	48.37%	40.63%	87.88%	12.12%
30	80.91%	48.20%	40.93%	87.93%	12.07%

Table D.35
Siamese Network: NSL-KDD Cup'99 One-Shot Confusion Matrix ($j = 5$) (Probe excluded from Training)

	Predicted Class					
Correct	Normal	DoS	Probe	R2L	U2R	Overall
Normal	5389 **(89.82%)**	89 *(1.48%)*	195 *(3.25%)*	245 *(4.08%)*	82 *(1.37%)*	75.31%
DoS	37 *(0.62%)*	5842 **(97.37%)**	95 *(1.58%)*	21 *(0.35%)*	5 *(0.08%)*	
Probe	1697 *(28.28%)*	2571 *(42.85%)*	565 **(9.42%)**	948 *(15.8%)*	219 *(3.65%)*	
R2L	54 *(0.9%)*	0 *(0%)*	55 *(0.92%)*	5800 **(96.67%)**	91 *(1.52%)*	
U2R	263 *(4.38%)*	0 *(0%)*	21 *(0.35%)*	720 *(12%)*	4996 **(83.27%)**	

Table D.36
Siamese Network: NSL-KDD One-Shot Accuracy (Probe excluded from Training) Using Different j Votes

No Votes	Overall	New Class (Probe)		Normal	
(j)	Accuracy	TPR	FNR	TNR	FPR
1	70.62%	18.80%	24.78%	77.53%	22.47%
5	75.31%	9.42%	28.28%	89.82%	10.18%
10	75.2%	4.83%	28.82%	91.08%	8.92%
15	75.12%	4.05%	29.08%	91.18%	8.82%
20	75.11%	3.47%	29.20%	91.45%	8.55%
25	75%	3.02%	29.55%	91.35%	8.65%
30	74.94%	2.68%	29.68%	91.33%	8.67%

Appendix E

Siamese Zero-Day Detection Results Tables

In Section 5.7, the proposed Siamese network model is evaluated on detecting unknown (zero-day) attacks based on pair similarity. Relying on the similarity, an attack is classified as unknown, if its similarity is less than a given threshold (i.e., the difference with all known classes is high). An attack class is excluded from training, one at a time, and used to mimic a zero-day attack. For completeness and transparency, the following subsections list the zero-day detection performance of the different attacks in the four datasets that are used for evaluation.

E.1 SCADA Dataset

Table E.1
Siamese Network: SCADA Zero-Day Accuracy (Wrong Connection excluded from Training) Using Different j Votes

No Votes	Overall Accuracy	Zero-Day Class (S14)		Benign Class		Known Attack Classes
	Classified As:					
(j)		Unknown	Normal	Normal	Unknown	Unknown
1	68.17%	99.5%	0.15%	13.15%	55.25%	15.44%
5	71.8%	99.55%	0.25%	14.25%	53.55%	12.16%
10	69.59%	99.95%	0.05%	5.5%	64.35%	15.63%
15	71.52%	99.9%	0.1%	6.8%	62.25%	13.21%
20	70.16%	99.95%	0.05%	3.65%	66.2%	15.06%
25	69.26%	99.95%	0.05%	2.1%	67.9%	16.3%
30	70.4%	99.9%	0.1%	2.75%	67.1%	14.84%

Table E.2
Siamese Network: SCADA Zero-Day Accuracy (Spoofing excluded from Training) Using Different j Votes

No Votes	Overall Accuracy	Zero-Day Class (S13)		Benign Class		Known Attack Classes
	Classified As:					
(j)		Unknown	Normal	Normal	Unknown	Unknown
1	62.78%	5.25%	16.55%	22.45%	12.6%	7.99%
5	64.49%	5.2%	22.05%	29.7%	11.1%	6.86%
10	63.8%	26.5%	14.2%	19.9%	28.75%	10.55%
15	64.15%	14.8%	18.35%	24.85%	20%	8.72%
20	64.14%	28.95%	13.35%	19.7%	30.8%	10.57%
25	64.18%	37.05%	11.25%	15.95%	39.45%	11.96%
30	64.15%	29.95%	13.35%	18.9%	33.15%	10.8%

Table E.3
Siamese Network: SCADA Zero-Day Accuracy (Sensor Failure excluded from Training) Using Different j Votes

No Votes	Overall Accuracy	Zero-Day Class (S12)		Benign Class		Known Attack Classes
	Classified As:					
(j)		Unknown	Normal	Normal	Unknown	Unknown
1	63.92%	48.45%	11.35%	17.45%	32.45%	9.18%
5	66.84%	43.15%	11.45%	20.2%	22.65%	6%
10	66.46%	55.45%	3.35%	10.25%	37.55%	8.75%
15	67.55%	47.35%	5.05%	11%	26.85%	6.21%
20	67.25%	52.45%	2.65%	6.45%	34.6%	7.43%
25	67.05%	55.6%	1.6%	4.4%	38.6%	8.15%
30	67.54%	51.25%	2.1%	4.9%	32.75%	6.71%

Table E.4
Siamese Network: SCADA Zero-Day Accuracy (Plastic Bag excluded from Training) Using Different j Votes

No Votes	Overall Accuracy	Zero-Day Class (S11)		Benign Class		Known Attack Classes
	Classified As:					
(j)		Unknown	Normal	Normal	Unknown	Unknown
1	71.54%	3.95%	22.05%	35.05%	13.55%	4.08%
5	73.75%	0.7%	28.25%	45.5%	4.45%	3.59%
10	73.61%	6.75%	24.4%	41.9%	7.85%	5.36%
15	74.03%	1.05%	26.8%	44.45%	2.4%	4.05%
20	73.75%	5.25%	25%	43.9%	3.55%	5.23%
25	73.81%	8.6%	24.25%	42.75%	5.4%	5.95%
30	73.7%	4.2%	24.7%	44.6%	2.45%	5.21%

Table E.5
Siamese Network: SCADA Zero-Day Accuracy (Person Hitting Low Intensity excluded from Training) Using Different j Votes

No Votes	Overall Accuracy	Zero-Day Class (S10)		Benign Class		Known Attack Classes
	Classified As:					
(j)		Unknown	Normal	Normal	Unknown	Unknown
1	54.41%	32.1%	3.4%	14.15%	13.25%	6.46%
5	57.83%	29.3%	1.55%	19.15%	13.6%	5.42%
10	48.79%	0%	0%	9.9%	29.5%	14.29%
15	58.76%	45.5%	0%	13.25%	21.85%	6.73%
20	58.65%	58.45%	0%	7.45%	30%	8.66%
25	58.37%	63.95%	0%	5.5%	36.1%	10.38%
30	58.8%	59.8%	0%	7%	31.25%	8.38%

Table E.6
Siamese Network: SCADA Zero-Day Accuracy (Person Hitting Medium Intensity excluded from Training) Using Different j Votes

No Votes	Overall Accuracy	Zero-Day Class (S9)		Benign Class		Known Attack Classes
	Classified As:					
(j)		Unknown	Normal	Normal	Unknown	Unknown
1	52.06%	49.15%	1.7%	11.65%	22.9%	15.55%
5	53.94%	25.6%	0.55%	11.15%	16.65%	12.1%
10	52.15%	31%	0%	4.6%	28.9%	19.19%
15	53.91%	15.85%	0%	4.3%	19.75%	13.81%
20	52.66%	18.45%	0%	2.3%	25.2%	17.21%
25	51.84%	19.8%	0%	1.15%	28.65%	19.88%
30	52.76%	11.75%	0%	1.5%	22.5%	16.38%

Table E.7
Siamese Network: SCADA Zero-Day Accuracy (7 Floating Object excluded from Training) Using Different j Votes

No Votes	Overall Accuracy	Zero-Day Class (S7)		Benign Class		Known Attack Classes
	Classified As:					
(j)		Unknown	Normal	Normal	Unknown	Unknown
1	79.26%	58.55%	1.6%	43.8%	7.3%	0.81%
5	82.11%	58.55%	1%	61.2%	1.55%	0.18%
10	81.96%	58.85%	0.65%	61%	8.2%	0.78%
15	83.08%	58.45%	0.7%	70.1%	0.75%	0.12%
20	83.09%	58.65%	0.6%	70.75%	2.95%	0.35%
25	83.11%	58.75%	0.6%	70.7%	5.35%	0.6%
30	83.49%	58.45%	0.7%	73.95%	1.55%	0.2%

Table E.8
Siamese Network: SCADA Zero-Day Accuracy (2 Floating Objects excluded from Training) Using Different j Votes

No Votes	Overall Accuracy	Zero-Day Class (S6)		Benign Class		Known Attack Classes
	Classified As:					
(j)		Unknown	Normal	Normal	Unknown	Unknown
1	67.63%	0%	0%	38.5%	8.15%	4.55%
5	71.08%	0%	0%	56.85%	3.8%	3.53%
10	70.36%	0%	0%	51.8%	12.45%	4.83%
15	71.43%	0%	0%	59.3%	3.35%	3.54%
20	71.08%	0%	0%	58.35%	6.45%	3.97%
25	70.81%	0%	0%	57%	10.2%	4.3%
30	71.17%	0%	0%	61%	5.05%	3.69%

Table E.9
Siamese Network: SCADA Zero-Day Accuracy (Humidity excluded from Training) Using Different j Votes

No Votes	Overall Accuracy	Zero-Day Class (S5)		Benign Class		Known Attack Classes
	Classified As:					
(j)		Unknown	Normal	Normal	Unknown	Unknown
1	52.6%	26.25%	11%	12.35%	18.35%	12.85%
5	55.92%	24.25%	7.9%	11.45%	10.75%	9.78%
10	56.94%	52.2%	1.25%	3.95%	20.55%	16.2%
15	58.47%	37.05%	1.4%	4.1%	10%	11.05%
20	58.79%	53.6%	0.4%	2.3%	14.45%	13.64%
25	58.75%	64.5%	0%	1.6%	19.05%	15.83%
30	59.54%	54.55%	0.05%	1.25%	13.05%	12.45%

Table E.10
Siamese Network: SCADA Zero-Day Accuracy (Blocked Measure 2 excluded from Training) Using Different j Votes

No Votes	Overall Accuracy	Zero-Day Class (S3)		Benign Class		Known Attack Classes
	Classified As:					
(j)		Unknown	Normal	Normal	Unknown	Unknown
1	68.69%	0%	0%	42.55%	10.25%	0.86%
5	71.8%	0%	0%	55.6%	3.4%	0.24%
10	72.48%	0%	0%	54.9%	8.9%	1.31%
15	73.43%	0%	0%	61.05%	2.3%	0.28%
20	73.54%	0%	0%	61.6%	4.3%	0.65%
25	73.59%	0%	0%	61%	5.5%	0.87%
30	73.96%	0%	0%	64.6%	1.7%	0.34%

Table E.11
Siamese Network: SCADA Zero-Day Accuracy (Blocked Measure 1 excluded from Training) Using Different j Votes

No Votes	Overall Accuracy	Zero-Day Class (S2)		Benign Class		Known Attack Classes
	Classified As:					
(j)		Unknown	Normal	Normal	Unknown	Unknown
1	69.38%	0%	0%	40.65%	8.55%	4.72%
5	72.33%	0%	0%	59.55%	1.6%	4.07%
10	72.3%	0%	0%	59.7%	6.7%	5.32%
15	73.45%	0%	0%	66.95%	0.85%	4.38%
20	73.18%	0%	0%	66.35%	3.15%	4.88%
25	72.97%	0%	0%	64.95%	4.85%	5.18%
30	73.41%	0%	0%	68.6%	1.35%	4.78%

E.2 CICIDS2017 Dataset

Table E.12
Siamese Network: CICIDS2017 Zero-Day Accuracy (FTP excluded from Training) Using Different j Votes

No Votes	Overall Accuracy	Zero-Day Class (FTP)		Benign Class		Known Attack Classes
	Classified As:					
(j)		Unknown	Normal	Normal	Unknown	Unknown
1	56.57%	31.82%	3.75%	38.43%	53.3%	23.09%
5	64.64%	11.48%	4.83%	61.92%	32.93%	10.34%
10	64.22%	12.27%	3.62%	58.88%	37.73%	11.59%
15	65.63%	10.22%	4.87%	67.23%	29.33%	10.76%
20	65.39%	11.38%	4.02%	65.03%	31.8%	11.31%
25	65.19%	11.67%	3.77%	63.82%	33.15%	11.63%
30	65.84%	11.03%	4.35%	67.6%	29.37%	11.24%

Table E.13
Siamese Network: CICIDS2017 Zero-Day Accuracy (DoS (Slowloris) excluded from Training) Using Different j Votes

No Votes	Overall Accuracy	Zero-Day Class (DoS (Slowloris))		Benign Class		Known Attack Classes
	Classified As:					
(j)		Unknown	Normal	Normal	Unknown	Unknown
1	67.45%	71.65%	6.33%	49.87%	41.8%	21.54%
5	80.56%	76.28%	4.35%	73.8%	21.17%	8.82%
10	82.46%	87.58%	2%	72.97%	24.32%	10.94%
15	83.61%	86.72%	2.52%	79.2%	18.08%	10.17%
20	83.53%	88.57%	2.18%	77.6%	20.1%	11.02%
25	83.59%	89.52%	1.92%	77.02%	20.78%	11.34%
30	83.95%	88.88%	2.12%	79.48%	18.28%	11.04%

E.3 KDD Cup'99 Dataset

Table E.14
Siamese Network: KDD Cup'99 Zero-Day Accuracy (U2R excluded from Training) Using Different j Votes

No Votes	Overall Accuracy	Zero-Day Class (U2R)		Benign Class		Known Attack Classes
	Classified As:					
(j)		Unknown	Normal	Normal	Unknown	Unknown
1	60.18%	52.07%	2.62%	46.3%	46.1%	30.18%
5	71.35%	50.07%	0.13%	64.93%	26.9%	18.87%
10	72.01%	50.22%	0%	66.28%	25.63%	18.31%
15	72.51%	50.22%	0%	67.4%	24.52%	17.84%
20	72.47%	50.22%	0%	67.3%	24.62%	17.87%
25	72.5%	50.22%	0%	67.37%	24.55%	17.84%
30	72.52%	50.22%	0%	67.43%	24.48%	17.83%

Table E.15
Siamese Network: KDD Cup'99 Zero-Day Accuracy (Probe excluded from Training) Using Different j Votes

No Votes	Overall Accuracy	Zero-Day Class (Probe)		Benign Class		Known Attack Classes
	Classified As:					
(j)		Unknown	Normal	Normal	Unknown	Unknown
1	64.51%	55.38%	6.27%	53.63%	38.53%	25.82%
5	75.1%	53.17%	7.32%	71.1%	18.93%	16.04%
10	75.2%	53.43%	7.38%	71.93%	18.13%	16.35%
15	75.6%	53.35%	7.48%	72.53%	17.5%	15.85%
20	75.52%	53.47%	7.47%	72.5%	17.57%	16.01%
25	75.48%	53.48%	7.47%	72.47%	17.62%	16.07%
30	75.51%	53.47%	7.48%	72.52%	17.55%	16.03%

Table E.16
Siamese Network: KDD Cup'99 Zero-Day Accuracy (DoS excluded from Training) Using Different j Votes

No Votes	Overall Accuracy	Zero-Day Class (DoS)		Benign Class		Known Attack Classes
	Classified As:					
(j)		Unknown	Normal	Normal	Unknown	Unknown
1	57.05%	75.15%	0.42%	45.32%	46.5%	38.06%
5	70.85%	68%	0.07%	64.15%	26.02%	23.04%
10	71.25%	67.65%	0.07%	65.38%	24.88%	22.77%
15	71.95%	67.4%	0.07%	66.85%	23.42%	21.97%
20	71.9%	67.38%	0.07%	66.83%	23.43%	22.03%
25	71.92%	67.38%	0.07%	66.9%	23.37%	22.02%
30	71.93%	67.38%	0.07%	66.93%	23.33%	22.01%

E.4 NSL-KDD Dataset

Table E.17
Siamese Network: NSL-KDD Zero-Day Accuracy (U2R excluded from Training) Using Different j Votes

No Votes	Overall Accuracy	Zero-Day Class (U2R)		Benign Class		Known Attack Classes
	Classified As:					
(j)		Unknown	Normal	Normal	Unknown	Unknown
1	54.16%	71.2%	21.82%	36.67%	59.95%	43.92%
5	66.52%	58.18%	36.4%	58.27%	39.8%	25.18%
10	67.17%	61.67%	34.38%	56.7%	42.65%	24.89%
15	68.52%	53.17%	42.97%	67.95%	31.37%	22.99%
20	68.5%	54.7%	41.58%	67.12%	32.23%	23.43%
25	68.2%	55.28%	41.02%	65.6%	33.77%	23.59%
30	68.76%	50.98%	45.32%	71.3%	28.05%	22.79%

Table E.18
Siamese Network: NSL-KDD Zero-Day Accuracy (Probe excluded from Training) Using Different j Votes

No Votes	Overall Accuracy	Zero-Day Class (Probe)		Benign Class		Known Attack Classes
	Classified As:					
(j)		Unknown	Normal	Normal	Unknown	Unknown
1	61.32%	58.57%	13%	58.78%	39.8%	34.82%
5	70.01%	52.72%	16.8%	76.3%	22.55%	24.52%
10	70.08%	52.5%	17.1%	76.85%	22.03%	24.62%
15	71.31%	52.28%	17.27%	77.15%	21.72%	22.56%
20	71.27%	52.28%	17.25%	77.12%	21.77%	22.62%
25	71.32%	52.38%	17.25%	77.08%	21.8%	22.57%
30	71.59%	52.33%	17.27%	77.08%	21.8%	22.08%

Table E.19
Siamese Network: NSL-KDD Zero-Day Accuracy (DoS excluded from Training) Using Different j Votes

No Votes	Overall Accuracy	Zero-Day Class (DoS)		Benign Class		Known Attack Classes
	Classified As:					
(j)		Unknown	Normal	Normal	Unknown	Unknown
1	52.48%	19.12%	6.1%	64.17%	33.8%	38.43%
5	61.63%	3.9%	7.58%	79.07%	19.75%	23.17%
10	61.82%	3.78%	7.48%	79.52%	19.37%	22.97%
15	62.31%	3.48%	7.78%	79.6%	19.27%	22.05%
20	62.33%	3.45%	7.82%	79.62%	19.25%	22%
25	62.36%	3.52%	7.77%	79.63%	19.23%	21.98%
30	62.38%	3.38%	7.88%	79.63%	19.23%	21.91%

Appendix F

Autoencoder Experiment ROC Plots

In Chapter 6, an autoencoder model is evaluated on its ability to detect zero-day attacks. Figure F.1 shows the ROC curves for each of the attacks in the CICIDS2017 dataset, based on the results discussed in Section 6.5.1.

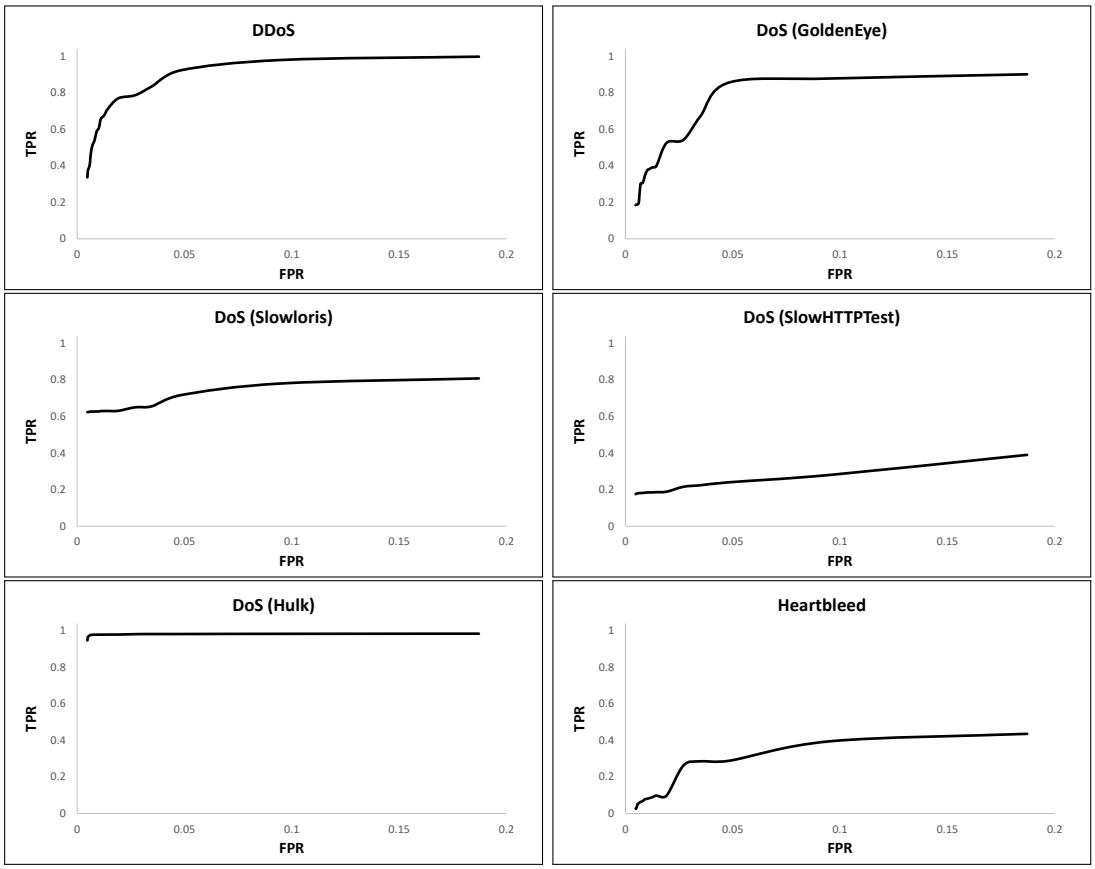


Figure F.1
Autoencoder Classification ROC Curves

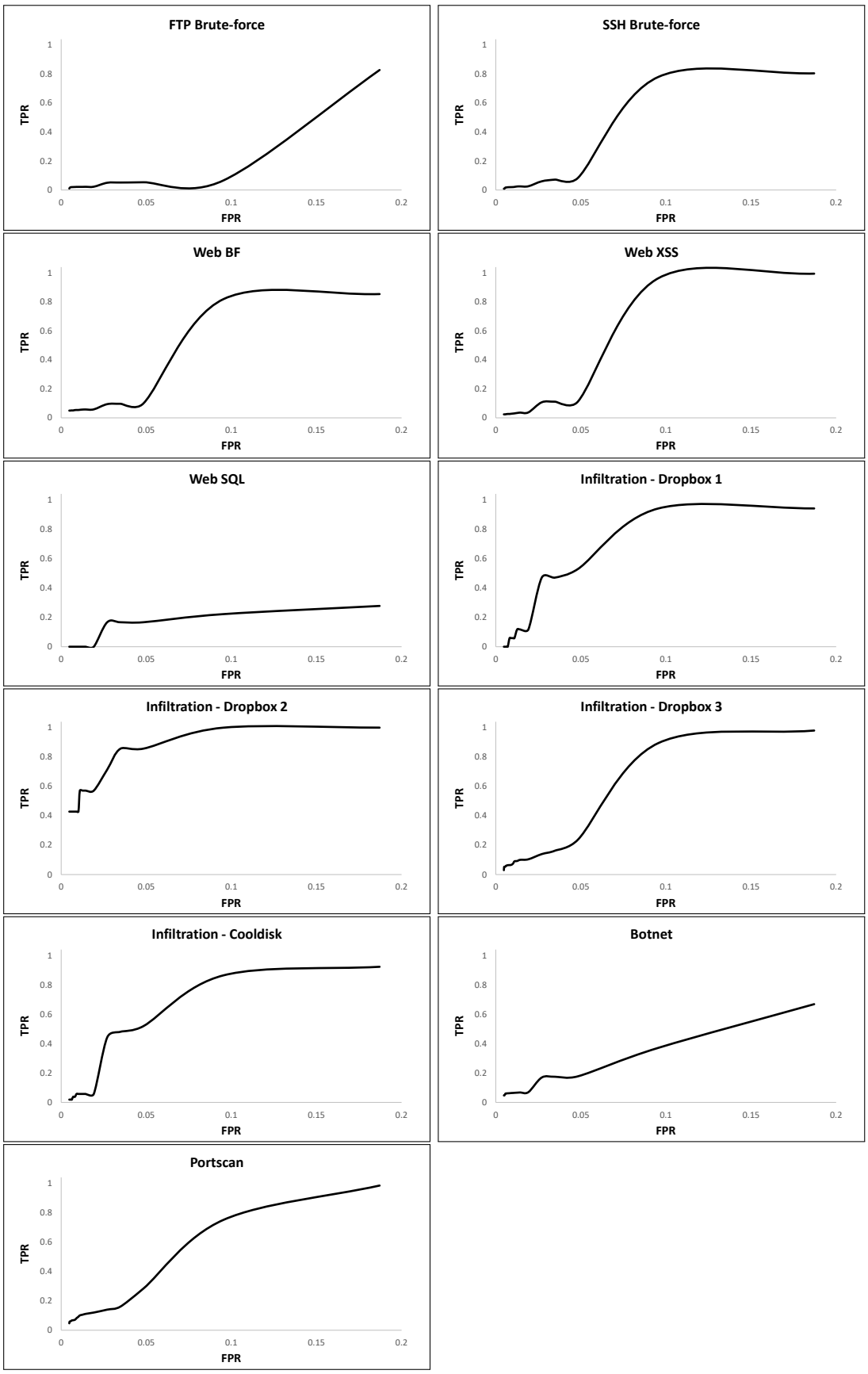


Figure F.1
Autoencoder Classification ROC Curves