Heriot-Watt University

Masters Thesis

INTRUSION DETECTION USING ARTIFICIAL INTELLIGENCE

Author:
Surya Vinayak Jyothish
(H00343483)

Supervisor:
Dr. Hani Ragab

A thesis submitted in fulfilment of the requirements for the degree of MSc. Network Security

in the

School of Mathematical and Computer Sciences

December 2021

Declaration of Authorship

I, Surya Vinayak Jyothish, declare that this thesis titled, 'INTRUSION DETECTION USING ARTIFICIAL INTELLIGENCE' and the work presented in it is my own. I confirm that this work submitted for assessment is my own and is expressed in my own words. Any uses made within it of the works of other authors in any form (e.g., ideas, equations, figures, text, tables, programs) are properly acknowledged at any point of their use. A list of the references employed is included.

Signed: Surya Vinayak Jyothish

Date: 08-11-2021

"Everything is theoretically impossible, until it is done."

Robert A. Heinlein.

"The Internet is the crime scene of the 21st Century."

Manhattan District Attorney Cyrus Vance Jr., October 2010

Abstract

The aim of this project is to apply a combination of deep learning and machine learning algorithms for intrusion detection. The term Artificial intelligence is used to broadly describe machine learning techniques that can imitate human behavior when trained. The deep learning techniques that will be applied are DNN (Deep Neural Networks), RNN (Recurrent Neural Network), DBN (Deep Belief Network), CNN (Convolutional Neural Network). The shallow machine learning techniques that will be applied are DT (Decision Tree), SVM (Support Vector Machine), KNN (K Nearest Neighbors). The project will include evaluation of feature selection methods CFS (Correlation-based Feature Selection), IG (Information Gain), AR (Attribute Ratio), and Mutual Information. The project will evaluate models on the dataset CICIDS2017 and CSE-CIC-IDS2018. The model will try to experiment with more deep learning such as MLP (Multi-Layer Perceptron) and Autoencoders and try to experiment with combining different deep learning techniques.

Acknowledgements

I would like to thank Professor Ragab for supervising, helping, and guiding me throughout this dissertation. I would like to thank my parents for their love and support. I would also like to thank my friends for their help and support.

Contents

List of Figures

List of Tables

Abbreviations

Anomaly detection systems – ADS

Area Under the Curve - AUC

Area Under the Receiver Operating Characteristics – AUROC

Artificial Neural Network – ANN

Attribute Ratio - AR

Common Vulnerabilities and Exposures project – CVE

Confidentiality, integrity, and availability - CIA

Convolutional Neural Network – CNN

Correlation-based Feature Selection – CFS

Decision Tree – DT

Deep Belief Network – DBN

Deep Learning – DL

Defense Advanced Research Projects Agency – DARPA

Denial of Service - DoS

DNS over HTTPS – DoH

False Negative – FN

False Positive – FP

False-negative rate – FNR

Frequent Episode Rule – FER

Gain ratio – GR

Gated Recurrent Unit – GRU

Host-Based IDSs – HIDS

Information gain – IG

Internet Control Message Protocol – ICMP

Internet Protocol – IP

Intrusion detection and prevention systems – IDPS

Intrusion Detection System – IDS

K-nearest Neighbors algorithm – KNN

Least-squares support-vector machine – LSSVM

Long Short-term Memory – LSTM

Machine Learning – ML

Multi-layer Perceptron – MLP

Naive Bayes algorithm – NB

Network-Based IDSs – NIDS

Non-symmetric Deep Auto-Encoder – NDAE

Open Systems Interconnection model – OSI

Operating System – OS

Packet Capture – PCAP

Random Forest – RF

Receiver Operating Characteristics – ROC

Recurrent Neural Network – RNN

Remote to Local - R2L

Restricted Boltzmann Machine – RBM

Self- taught Learning – STL

State-space model – SM

Support Vector Machine - SVM

Transmission Control Protocol – TCP

True Negative – TN

True negative rate – TNR

True Positive – TP

True positive rate – TPR

United States – US

User Datagram Protocol – UDP

User to Root - U2R

Virtual Machines – VM

Chapter 1

Introduction

Technology has become an integral part of our daily lives. There has been an increase in the number of devices being used and connected to the internet. The increasing popularity of IoT has also led to this increase in devices. The estimate for the number of connected devices in 2020 was 20 billion [Doshi et al., 2018]. Devices are connected using networks that allow them to communicate with each other. The internet is the largest network connecting billions of devices.

With the rising number of devices and networks, there is a rise in security threats and vulnerabilities. People with knowledge of these threats and vulnerabilities can use them to obtain information to gain control of devices. Hence it is important to understand intrusion, intrusion detection, and intrusion detection system (IDSs). Understanding the requirements and challenges of current IDS technology and research is also vital to model an IDS. Since 1980, researchers have worked on providing security solutions that can protect the confidentiality, integrity, and availability (CIA triad) of the services [Anderson, 1980]. In the current age where data is valued, the protection of data is a fundamental right. The improvement in technology also leads to improved methods to exploit device vulnerabilities. These vulnerabilities also increase with improved technology. The number of cyber-attacks have increased in 2020. Cisco has estimated the number of cyber-attacks to rise to 15.4 million by 2023. Cyber-crimes can cause damage and losses to people, companies, and organizations. Sensitive data can be lost, collected, and misused. The risk of these cyber-attack demands good detection and prevention methods.

The detection of attacks is vital to prevent or minimize the impact of the attack. There are many security solutions available to prevent attacks from occurring such as firewalls and network access systems. It is important to detect attacks and take protective measures. IDS are used to detect an attack.

It is impossible to prevent all network attacks. Early IDS research was based on defining sets of rules. These rule-based IDS were able to detect existing attacks but were not detect new or zero-day attacks. With the rising popularity of ML and AI concepts, researchers have shifted their attention to using ML techniques to detect network intrusion. ML techniques can detect patterns in recorded attacks to detect anomalous behavior in new or unrecorded attacks.

1.1 Aim

The aim of this thesis is to evaluate the effectiveness of Traffic grouping net-flow packets records based on the network attack to detect and classify network attacks and anomalies, and to apply different Machine Learning algorithms to evaluate the effectiveness of this approach.

1.2 Objectives

1.2.1 Main Objectives

In this MSc project, there are three objectives. The primary objective is to group net-flow packet data of the carried out network attacks in the dataset. The thesis focus on the applicability of grouping individual attack net-flow record as a single record. A single net-flow record can be considered as a single line of communication between the devices. The aim is to find patterns within the whole communication that can help detect and classify the attacks. The features in the dataset can be extracted to form super features. Use Feature selection methods to identify important features in the data. After this, the net-flow data will be grouped based on the type of attack. A labeled net-flow dataset will contain the net-flow communication between devices. For an attack, the net-flow packets related to that attack will contain patterns that can be useful for detecting that

network attack. Machine learning can detect and classify attacks based on the found patterns. The second objective is to use various shallow and deep machine learning algorithms to evaluate the proposed approach.

Deep machine learning models can be used to extract features or encode the data. This approach clusters all the net-flow packet data based on the type of attack. Deep learning techniques can be used to extract hidden patterns within the data entries. This pattern recognition is used widely in image and speech recognition and modeling. Similarly, this grouped net-flow data will have hidden patterns that can be extracted. The third aim is to use deep learning techniques to extract hidden features from the proposed data.

The summary of the main objectives are:

• Select a labelled net-flow dataset. Extract superfeatures from the dataset.

• Implement and evaluate feature selection algorithms.

• Implement and evaluate the proposed NetFlow grouping approach.

• Implement and evaluate shallow and deep machine learning algorithms.

• Implement and evaluate various combinations of deep learning algorithms and shallow machine learning algorithms.

• Compares the results of the various models created using different feature subsets.

1.2.2 Optional Objectives

• Implement and evaluate various combinations of deep learning algorithms.

• Evaluate the performance of best-performing models in a real-world network environment.

1.2.3 Expected Results

• The extracted super features will improve the performance of the designed ML model.

• After feature selection, the resultant subset will provide better results compared to the entire feature set.

• The proposed approach will perform better on deep learning algorithms compared to shallow learning methods.

• The use of deep learning algorithms to extract hidden patterns will improve the performance of shallow learning methods.

• The use of deep learning algorithms to extract hidden patterns will result in no improvements to the performance of deep learning methods.

1.3 Research Questions

Some of the research questions we hope to answer in this study are:

• How effective is the proposed Netflow grouping of individual attacks in intrusion detection?

• How effective is the use of shallow and deep learning models in intrusion detection?

• How effective is the use of deep learning algorithms as feature extractors?

• What is the importance of IP address features in ML-based intrusion detection modules and, how does it affect the performance of the IDS?

• What is the importance of TCP flags as features for ML-based intrusion detection modules?

• What are the important Net-flow features used for building ML-based intrusion detection models?

• How effective and feasible is the implementation of Traffic Grouping Based intrusion detection in a real-world scenario?

Chapter 2

Literature Review

2.1 Intrusion

Bace [2000] defined an intrusion as any act that compromises the confidentiality, integrity, and availability (CIA triad) of the network or system. Intrusions can be classified based on the effect they have on the network or device. Passive attacks are attacks where the attacker only views or snoops around the network without changing the state of the entity. Examples of common passive attacks are traffic analysis and packet sniffing. Active attacks result in a comprise or change in the state of the device of the system. Attacks such as Denial of Service or masquerading are types of active attacks.

A widely accepted and approved taxonomy is considered good. The taxonomy should account for all types of attacks. A complete overview of intrusions in taxonomy is hard to achieve. If a taxonomy can provide classification criteria for the attacks, it is considered complete. Hansman and Hunt [2005] mentions that a taxonomy understood by anyone related or interested in the security field (comprehensive) is a good taxonomy. The taxonomy must explain the criteria and rules used for classification (unambiguous, determinism) and should be repeatable if performed again. The taxonomy should clearly define any terms used in it for the clarity of the reader. The categories in the taxonomy should be defined such that they are mutually exclusive.

The Protection Analysis (PA) [Bisbey, Richard and Hollingworth, 1978] and Research in Secured Operating Systems (RI-SOS) [Abbott et al., 1976] were few of the earliest taxonomy related to intrusion detection. These taxonomies focused on the classification

of vulnerabilities rather than intrusions. These taxonomies provide background when proposing new taxonomies.

Kendall et al. [Cunningham et al., 1999, Kendall, 1999] classified network attacks into four categories in her proposed taxonomy. These categories are Denial of Service (DoS), Remote to Local (R2L), User to Root (U2R), and Probing. Kendall et al. formed these categories by analyzing the 1998 DARPA intrusion dataset, sponsored by MIT Lincoln Library.

Hansman and Hunt [2005] proposed a taxonomy for network attacks that discuss the vulnerabilities and targets of attack. Hansman split network attacks into four categories with multiple stages. The categories were defined using the attack vector, the target of attack, vulnerabilities or exploits used to perform the attack, and the effect of the attack payload on the target. Other categories were mentioned, such as the amount of damage done or how the attack can be defended against but not as in-depth as the first four discussed.

Hindy et al. [2020] proposed a taxonomy that gives a broad overview of intrusion detection attacks in the last decade, categorizing threats based on the source, type of threat (active or passive), and OSI layer affected by the threat. Hindy et al. also categorize threats based on network, Host, Software, Physical and Human threats. Monitoring the flow or number of packets sent over a network can be used to detect Network Threats. Hindy et al. also briefly discusses classification based on attack tools.

2.2 Intrusion Detection

Intrusion detection is defined as the process of monitoring and analyzing a network or device for security threats, vulnerabilities, and misuse (broadly called intrusions) [Azad, 2008, Roberto Di Pietro and Luigi V. Mancini, 2008]. The process of detecting an intrusion is important to perform preventive or corrective measures when an intrusion occurs.

Intrusion detection is vital to prevent or minimize the harm that an attack does to the network or system by detecting the intrusion. Intrusion detection can be performed in two ways, using anomaly detection or signature detection. Signature Detection (also

known as misuse detection) compares patterns and knowledge of the previous attacks to detect those attacks. These patterns are called signatures. These intrusion signatures are saved in a dataset. Signature detection is an effective way to detect already known attacks. Anomaly detection compares the behavior of a network or device with a predefined normal behavior or rules (looks for suspicious behavior). If suspicious behavior is detected, the alarm is raised for an intrusion. This method is effective in detecting unknown or attacks that have varied (changing signatures) patterns. Therefore, this detection method can raise an alert and prevent the system or network from being compromised. This normal behavior may be learned using Machine Learning (ML) techniques or defined by users or administrators.

Anomaly-Based Detection	Signature-Based Detection
Pro	
• Simple and effective method to prevent known attacks. • Analysis is performed using contextual analysis.	• Effective in detecting unknown or varied attacks and unforeseen vulnerabilities. • Less Depended on OS. • Can be used to detect privilege abuse.
Con	
• Ineffective in detecting unknown or varied attacks. • Analysis does not use states and protocols. • It is difficult and time-consuming to maintain up-to-date signatures and patterns.	• Weak accuracy profiles due to changing observed events • Unavailable during building profiles. • Difficult to raise alerts in real-time.

TABLE 2.1: DIFFERENCE BETWEEN ANOMALY-BASED AND SIGNATURE-BASED INTRUSION DETECTION [Liao et al., 2012]

There have been many research and studies conducted for ML-based intrusion detection models. Hindy et al. [2020] had conducted a literature survey with prominent IDS papers in the range (2008 – 2020). The survey records the intrusion detection model used with the dataset used to train the model and the parameters used to evaluate the model.

Liao et al. [2012] surveyed the intrusion models research and classified them on the types of the detection approaches. The models were classified based on the approach. The classifications were statistical-based, pattern-based, rule-based, state-based, and heuristic-based detection approaches. The survey also mentions the instruction detection methodology, types of attack detected (unknown or known attacks), their source data type, and performance of the model.

Buczak and Guven [2016] surveyed the various ML and DL techniques used in intrusion detection. The survey details the intrusion detection approaches. The survey also gives the complexity of ML-based intrusion detection models.

2.3 Intrusion Detection System

An Intrusion Detection System (IDS) can be defined as a software application or device that monitors, decides, and reports a network or system for policy violations and malicious and suspicious activities [Jabez and Muthukumar, 2015]. IDS aims to detect an intrusion before the system or network is harmed or compromised. FIG 1 shows a general IDS structure. The solid lines show how data flows while dash lines show the response to intrusions [Wu and Banzhaf, 2010].

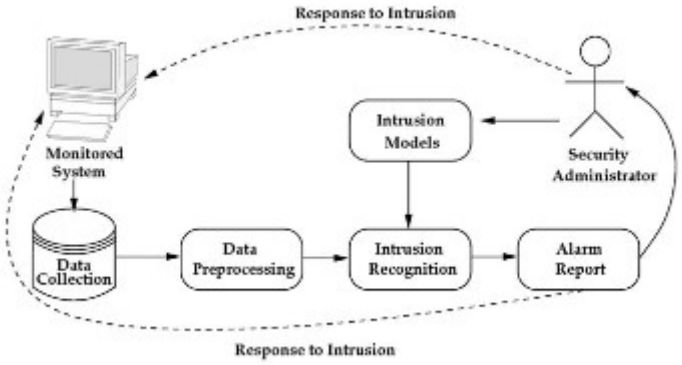


FIGURE 2.1: GENERAL IDS STRUCTURE [Wu and Banzhaf, 2010]

An IDS monitors a system and obtains data such as traffic flow, data packets, and log files that can help detect intrusions. A pre-processing module process the data and forms patterns or signatures. This pattern is forwarded to a recognition module where a dataset of known attack signatures. The signature is searched in the dataset. The system administrator is notified if an intrusion is detected and an alarm is triggered. The system administrator proceeds to decide on what action to take on the intrusion.

IDS can be classified based on the detection method used, response to detected intrusions, locus of detection, and audit data sources. IDS classification based on Audit Data Source:

• Network intrusion detection systems (NIDS): NIDS monitors and analyses traffic that flows to and from the devices in the network and compares the traffic to any known attacks. An alert is raised when an attack or suspicious behavior is detected. These systems are also capable of dropping harmful data packets to protect the network.

• Host intrusion detection systems (HIDS): HIDS are run on individual devices in a network. HIDS monitors the flow of data packets through the device. They take snapshots of system files for comparisons. When the HIDS detects the modification or deletion of system files or an attack and suspicious behavior, an alert is triggered.

Network-Based IDS	Signature-Based Detection
Pro	
• Can monitor large networks with good IDS placement. •Has little impact on the network and is hard to detect by attackers.	• Attacks within the host can be detected. NIDS cannot detect these types of attacks • Can detect intrusions in encrypted traffic. • Can detect intrusions such as Trojan during OS audit trials.
Con	
• Has a hard time monitoring and detecting intrusion in high traffic or encrypted networks. • NIDS can only detect attack initiation but cannot detect whether the attack was a success or failure • Has problems dealing with fragmented packet attacks.	• Hard to manage and perform. Specifications depend on the host device. • The IDS can be attacked and disabled in an attack targeting the information and analysis sources that are on the host device (e.g., DoS targeting IDS).

TABLE 2.2: DIFFERENCE BETWEEN NIDS AND HIDS [Bace and Mell, 2001]

IDS classification based on the method of detection:

• Signature-based or Misuse-based IDS: This method detects threats using specific patterns. These patterns include network traffic, file signature, or file contents. The obtained signature is compared with a library containing known attack signatures and patterns. When a match is detected, an alert is triggered. This method can easily detect known attacks. This method has difficulty in detecting attacks where patterns are unavailable (new) or vary drastically.

• Anomaly-based IDS: This method uses machine learning to detect attacks or threats. The IDS looks for unusual activity or actions to raise an alert. Unknown attacks are

detected using this method. The IDS monitors system activities and configurations to detect suspicious activities.

The Signature-based IDS are better at detecting known attacks. Anomaly-based detection is better at detecting unknown attacks. Anomaly-based detection is slower as a pattern needs to be built for the unknown attack. Current IDS uses Anomaly-based detection if the attack is unknown and saves it to a library which can later be used by signature-based methods. These IDS are called Hybrid Intrusion detection systems [Patel and Buddhadev, 2012].

IDS classification based on action taken:

• Active IDS: These IDS take actions to quarantine affected networks and devices without any actions from the system administrators. These IDS are also called Intrusion detection and prevention systems (IDPS). IDPS detects intrusions automatically and performs preventive measures to prevent damages or compromise to the system or network.

• Passive IDS: These IDS monitor the system or network. An alert is sent to the administrator when an intrusion is detected. The administrator decides on the measure to be taken against the attack. Passive IDS are not capable of deciding the preventive measures to be taken for the detected intrusion.

2.4 Requirements of an IDS

Intrusion prevention mechanisms such as firewalls, user authentication, and information protection are good first lines of defense to prevent intrusion [Gil et al., 2013, Lee and Stolfo, 1998]. These are good preventive measures but cannot completely stop attacks. When an intrusion does occur, there needs to detection and response mechanism to prevent any damage from the attack. An IDS requires to preserve confidentiality, integrity, and availability of the system, network, or service.

A reliable computer system or network includes confidentiality, integrity, and availability as a security requirement [Kumar, D Ashok and Venugopalan, 2017].Denning and Neumann [1985] defined a few requirements that an IDS, are:

• Detect Intrusions: An IDS should detect various types of intrusions that risk the CIA triad of security.

• Adaptability: An IDS should operate in any hardware, operating system, and application environment.

• Discriminating power: IDS should detect intrusions with high accuracy (high detection rate with low-rate false alarms).

• Ease of Use: The IDS should be user-friendly. The IDS can help users by prompting facilities, confirmations for irreversible actions.

• Modifiability: IDS should have options to change and add settings and rules without causing inconsistencies that lead to unexpected results. The system administrator must be given control to determine and adjust the discriminating power of the IDS.

• Self-Learning: IDS should learn normal behavior from monitoring and analyzing the system or network.

• Real-time detection: The audit records must be readily available to IDS for processing in real-time.

• Security of IDS and its database: The IDS system must enforce the privacy and integrity of the IDS database. The IDS should not also be vulnerable to DoS attacks.

2.5 Metrics for Evaluation of an IDS

IDS have different metrics that can be used to measure different abilities of the IDS. Commonly used metrics to evaluate the performance of an IDS are false-positive and true-positive rates (detection accuracy). These two parameters describe the discriminating power of an IDS. An ideal IDS provides a high true-positive rate and a low false-positive rate. A high true-positive results in more alerts being raised. A low false-positive rate results in fewer false alarms.

Evaluation parameters for an IDS can be obtained from a Confusion Matrix. A confusion matrix is a matrix or table that compares the actual and predicted values. The Confusion matrix is a matrix as shown in Fig 3. The left column (y-plane) represents the actual

value. The top row (x-plane) represents the predicted value. This can be interchanged but will change the position of properties in the matrix.

- True Positive (TP): The number of intrusions classified as intrusions.
- True Negative (TN): The number of normal behavior classified as normal.
- False Negative (FN): The number of attacks classified as normal behavior.
- False Positive (FP): The number of normal behavior classified as intrusions.

	Predicted Negative	Predicted Positive
Actual Negative	TN	FP
Actual Positive	FN	TP

FIGURE 2.2: CONFUSION MATRIX

An IDS should detect intrusions (High TP) and identify normal events (High TN). The IDS should not classify intrusions as normal activity (FN) as this allows the intrusion to affect the system or network without raising an alert to the system administrator. Similarly, an IDS should not classify a normal activity as an intrusion (FP), as raising an alert for normal behavior is an inconvenience. The metrics that be calculated using these parameters that are:
Overall accuracy: This metric is the probability the model can correctly classify the event. The equation is:

$$OverallAccuracy = ((TP + TN))/((TP + TN + FP + FN)) \tag{2.1}$$

Sensitivity (True positive rate (TPR)): This metric measures the probability of classifying intrusions are intrusions. This metric is also called Recall.

$$Sensitivity = TP/((TP + FN)) \tag{2.2}$$

Specificity (True negative rate (TNR)): This metric measures the probability of classifying normal activity as normal. This metric is also called Selectivity.

$$Specificity = TN/((TN + FP)) \tag{2.3}$$

Fallout (False positive rate (FPR)): This metric measures the probability of classifying normal events as intrusions.

$$Fallout = FP/((TN + FP)) \tag{2.4}$$

Miss Rate (False-negative rate (FNR)): This metric measures the probability of classifying intrusions as normal activity.

$$MissRate = FN/((TP + FN)) \tag{2.5}$$

Precision: This metric measures the probability of actual intrusions from all the predicted intrusions.

$$Precision = TP/((TP + FP)) \tag{2.6}$$

These metrics calculate the effectiveness of an IDS. They define how accurately the IDS detects intrusions while giving information about chances for false alarms and undetected intrusions. An intrusion detection model is designed by making a trade-off between sensitivity and fallout [Hindy et al., 2020].

The ROC plot visualizes the trade-off between the TP and FP or sensitivity and specificity. The ROC graph is plotted with the false positive rate is on the X-axis and the true positive rate on the Y-axis. The area under the ROC curve (AUC) measures the ability of the classifier model to distinguish between the class types (Intrusion class). The Higher this value (¡0.9), the better the precision of classification between the classes.

When dealing with imbalanced datasets, the F1 score metric provides a more reliable measure of the performance of the IDS [Hindy et al., 2020].

$$F1 = 2TP/((2TP + FP + FN)) \tag{2.7}$$

According to Hindy et al. [2020], other metrics used for evaluating an IDS are CPU consumption, throughput, and power consumption. These metrics determine the effectiveness of the IDS on different hardware and high-speed networks. Buczak and Guven [2016] also mentions that other than the accuracy of the model, the training time of the model and time taken to detect unknown attacks are also important parameters to evaluate an intrusion detection model.

2.6 Issues and Challenges

There is a variety of available IDS. Each of these IDS implements different detection models and hence shows distinct preferences over certain classes of intrusions. The analysis of these IDS models shows some of the challenges that an IDS faces. These challenges are [Hindy et al., 2020]:

• Capture and analysis of network packets in real-time are difficult due to the increase in network speed and bandwidth.

• Anomaly detection has high false alarm rates (false positive and false negative).

• Designing a portable Intrusion detection model (used in multiple systems with different specifications) is difficult because each environment or system has different security specifications and requirements.

• An ML-based intrusion detection model is trained and tested using an intrusion detection dataset. The model may work well within that dataset but not the other datasets.

• Intrusion detection datasets cannot cover all the known attacks and intrusions. This results in a trained IDS that does not work effectively in a real-world environment.

• Designing a reliable ADS model is challenging as datasets with recent attacks or real network activity is unavailable.

• Designing a model for normal and malicious traffic is difficult as this behavior changes over time.

2.7 Intrusion Detection Datasets

With the changes and advancements in network technology, IDS research requires a dataset that can reflect these changes. Currently, there are no datasets available that can reflect these real network changes. This hampers the IDS research from being applicable for real-life and commercial situations.

A dataset must meet a few criteria when generated or used in the implementation of a model. Viegas et al. [2017] mentions the dataset must meet the following criteria:

- The dataset should include real-world network traffic events.
- The activity must show all changes brought about by the intrusion in the network and system (complete scenario or impact of intrusion).
- The dataset entries must be labeled correctly.
- The dataset should include different variations of intrusion and normal events.
- The dataset should be easily updatable and reproducible. Therefore researchers can improve and compare the dataset.
- The dataset should not contain any confidential data.

Sharafaldin et al. [2017] mentions that the dataset should come with documentation about the features in the dataset and data collection process used. Cordero et al. [2015] recommends that the IDS dataset must provide well-expressed data and balanced class distribution. The normal activity entries must be varied because normal traffic depends on the system and network requirements. Cordero et al. [2015, 2021] also mentions the dataset must remove any disturbances, deflects, and irregular data entries.

Hindy et al. [2020] conducted a literature survey analyzing the available datasets used for IDS research. The datasets are cataloged based on their domain (General Purpose, Special Purpose Networks, Mobile Application). Commonly used datasets for intrusion detection research are:

DARPA 98,99:One of the oldest used datasets used is the DARPA 98 and 99, created by Lincoln Laboratory of the MIT [Abt and Baier, 2016]. Kendall [1999] proposed one of the earliest taxonomies based on this dataset. This dataset has a lot of defects, such as the

attack-related network packets being provided with a time to live value [Cordero et al., 2021]. These defects can lead to a biased learning model that hampers the efficiency of the IDS.

KDDCUPP99: The KDDCUPP99 dataset, also known as KDD 99, is an old dataset that is used for anomaly-based intrusion detection models. This dataset is a collection of records of net-flow from the Defense Advanced Research Projects Agency (DARPA) 98 dataset. This dataset has 41 features. The KDD 99 is an outdated dataset and has multiple issues [Creech and Hu, 2013, Tavallaee et al., 2009]. The KDDCUPP99 dataset is outdated, has no exact definition for the attacks, and normal activity does not resemble real-world normal traffic.

NSL-KDD: To overcome the issues with the KDDCUPP99 dataset and to provide a dataset better suited for IDS research, Tavallaee et al. [2009] proposed the NSL-KDD dataset. This dataset has select records from the KDD 99 dataset. The NSL-KDD has removed redundant records in KDD 99 dataset. The NSL-KDD dataset has records for each difficult group to improve the accuracy of ML techniques used in intrusion detection systems. These records were selected based on the number of KDD entries.

CICIDS2017: CICIDS2017 {IDS 2017 — Datasets — Research — Canadian Institute for Cybersecurity — UNB, 2021}. is a recent and popular dataset being used for IDS research [Rosay et al., 2020, Singh Panwar, Shailesh and Raiwani, YP and Panwar, 2019]. This dataset meets the dataset evaluation criteria proposed by Gharib et al. [2017]. This dataset has records of many intrusions and the steps and processes involved with those intrusions (multi-stage attacks like HeartBleed). This dataset contains 80 features. The features are collected from two networks, for an attacker and victim. The normal behavior was defined records were defined by using the CIC-B-Profile system. This dataset has a high-class imbalance that can lead to a biased ML model. There are high volumes of data with missing values in the dataset that need to can affect the performance of the model [Panigrahi and Borah, 2018].

CSE-CIC-IDS2018: CSE-CIC-IDS2018 {IDS 2018 — Datasets — Research — Canadian Institute for Cybersecurity — UNB, 2021} is an updated dataset version of the CICIDS2017 dataset released by the Canadian Institute for Cybersecurity. This dataset has a similar structure to CICIDS2017 with the same classes. This dataset has more

attack records compared to CICIDS2017 but has a similar class imbalance.[Leevy and Khoshgoftaar, 2020].

2.8 Related Works

2.8.1 Feature Selection

The use of ML techniques has improved the performance of the IDS. The selection of features in a training dataset improves the performance of an IDS. Improving the effectiveness of the intrusion detection dataset is of high research value. This research includes making better datasets [Sharafaldin et al., 2018a] with more focus on standards, fine-tuning parameters of models to yield slight improvements [Ingre and Yadav, 2015], using ML techniques to enhance features [Alrawashdeh and Purdy, 2017].

The selection of features in a dataset is vital to the performance of IDS. Features selection that favors a class can cause the ML model to be biased to that class. The selection process should be able to provide the most accurate information for all classes. This is difficult as different features highlight different intrusions, such as payload, target, or protocol. Commonly used feature selection methods are correlation-based feature selection (CFS), Information gain (IG), and gain ratio (GR).

Chae et al. [2013] proposed a method of using the frequency and average of the records in the feature to select the most shared features among classes. This method is called Attribute Ratio (AR). This proposed feature selection method was tested on the NSL-KDD dataset using the J48 algorithm in Weka. The algorithm produced an accuracy of '99.794%' for '22' features from the NSL-KDD dataset. This was slightly greater compared to using all the features (99.763%). This was compared to other feature selection methods CFS (accuracy: 99.781% with 25 features), IG (accuracy: 99.781% with 23 features), and GR (accuracy: 99.794% with 19 features).

Amiri et al. [2011] proposed a feature selection method using mutual information. The KDDCUPP99 dataset and model using 5 LSSVM were used to verify the proposed feature selection method. This method showed superior results compared to models without feature selection. The use of the feature section also reduces the computational time for the models.

A feature selection method was proposed by Kunhare et al. [2020] using particle swarm optimization also yielded better results when compared with other models. The accuracy obtained by using this feature section method with a random forest classifier was '87.83%'. The TPR, TNR, FNR, and FPR rates were also better compared to the other models.

Farahani [2020] proposed a feature selection method using correlation analysis. This feature selection method selects features with high correlation to the class and low correlation with the other features. The feature selection process was repeated to obtain the desired number of features. The proposed feature selection method was evaluated using various ML algorithms. Farahani's comparative analysis found that using the proposed method improved the performance metrics of the classifiers. The ideal number of features for the KDDCUPP99 dataset was 20 features (DT classifier). The improvement in the performance of the classifier using the proposed feature selection method is due to the removal of redundant information from the features that lower the learning capability of the model.

Buczak and Guven [2016] provides the list of features in the packet-header for various communication protocols (IP, TCP, UDP, and ICMP). The paper details the features that can be obtained from TCP connections and the NetFlow packet header. This knowledge can improve the manual feature selection process for IDS research. The use of feature selection techniques can improve the overall performance of the IDS. Features can also be selected in a way to overcome inconsistency issues in the dataset, such as record imbalance to a certain extent.

2.8.2 Machine Learning Algorithms

"The study of AI, stemming from computer science, aims to create computer systems that can interpret information and behave like a human" [Moxley-Wyles et al., 2020].

Artificial Intelligence is the study of creating machines that can perceive information as humans do. This is a highly researched topic that has is being used in a variety of fields. AI is a broad term that describes machine learning methods but is used to denote the use of deep learning techniques. Cohen [2020] defined machine learning as the process of classification and prediction of data using computer algorithms that create models

that follow a set of rules. ML algorithms are divided based on the learning process. The categories are supervised, unsupervised and, semi-supervised learning. In supervised learning, the data is shared with the expected output when training the model. The model learns and corrects itself using the labeled data. Intrusion detection models widely use supervised machine learning. In unsupervised learning, the data is provided with no labels. The model will learn the patterns without an expected output and classify the data. Semi-supervised learning uses both labeled and unlabeled data to learn rules and patterns for the unlabeled data while knowing the expected classes. This paper will mainly focus on supervised Deep learning techniques.

There are surveys conducted for the use of ML techniques in intrusion detection search. These surveys help future research with implementations and excepted results of the model for references.

Ferrag et al. [2020] surveyed DL algorithms. This survey performs a comparative study on different Deep Learning models used in intrusion detection research using two new datasets CSE-CIC- IDS2018 dataset and the Bot-IoT dataset. The survey also gives a review of the available datasets used for IDS research. The authors use false alarm rate, accuracy, and detection rate as performance indicators for the models. The paper covers a vast number of papers on simple deep learning techniques.

Gamage and Samarabandu [2020] performed a comparative study of various models and defined a taxonomy for Deep learning techniques. The survey explores the uses of multiple DL techniques used for an intrusion detection model. The use of deep and shallow ML techniques in conjunction yields better classifications and performance parameters. The authors discuss the current issues with IDS research as well.

2.8.2.1 Deep Neural Network

An artificial neural network is a board class that includes all DNN models. ANN usually has one hidden layer whereas DNN has multiple hidden layers.

Tang et al. [2016] proposed a supervised DNN model using 3 hidden layers (12, 6, 3). The dataset used to train the model uses 6 features from the NSL-KDD dataset. The model performs binary classification with a learning parameter between 0.1 – 0.0001. The best model had an accuracy of 75.75 with a learning parameter of 0.001. The number of

features selected for this model may have led to the underfitting of the model. Tang et al. perform a comparative analysis with other shallow machine learning methods (Naive Bayes, SVM, Decision Tree) to show the effectiveness of ANN for intrusion detection.

Vinayakumar et al. [2019] proposed an IDS detection module using an artificial neural network (ANN). The best model is a DNN using 5 hidden layers with 1024 neurons, and the Relu linear activation function. The layers were fully connected and the classification layer connected using a sigmoidal activation function. The model was trained and tested on multiple datasets (NSL-KDD, CICIDS 2017, UNSW-NB15, Kyoto, WSN-DS). The number of neurons used in the input layer and output layer varied with the used dataset. The proposed DNN architecture had a high TPR and low FPR. The model had a detection accuracy of 78.55 for the NSL-KDD dataset and 95.6% for the CICIDS2017 dataset. The model shows better performance for binary classification compared to multi-class classification. This model shows the promising results of using deep learning techniques for intrusion detection. The accuracy of the model was higher when trained and tested using the CICIDS2017 dataset compared to the KDDCUPP99 dataset because of the issues in the KDDCUPP99 dataset.

Ingre and Yadav [2015] used fine-tuning hyperparameters and feature selection to improve the performance of an ANN-based intrusion detection model. The DNN hyperparameters varied were the number of hidden layers, the algorithm used for classification, and the number of epochs. The NSL-KDD dataset was used with subsets of 41 features and 29 features. The best model for binary classification was a DNN with 21 hidden layers, an epoch set to 117, and the Levenberg-Marquardt (LM) learning algorithm. This model used 29 features set. The model had an accuracy of 81.2 on the test dataset. The FPR of the model was 29.5% while the FNR of the model was 3.2%. The multi-class classification model used 23 hidden layers, with an epoch of 771 and, using BFGS quasi-Newton backpropagation learning algorithm. The model achieved an accuracy of 79.9%. For multi-class classification, the model was able to classify 'DoS' and 'Probe' attacks but had issues with 'U2R' and 'R2L' classes. The paper describes the importance of using feature selection. This low classification for two classes is due to the low number of records in the dataset that describe those classes.

Pawlicki et al. [2020] proposed a DNN model for IDS and Adversarial attacks detection. The model is a binary classifier trained using the CICIDS2017 dataset. The IDS classification DNN has 3 hidden layers (40,20,20) with the Relu activation function. The model has an accuracy of 98.27%. The model output was fed to an adversarial attack detection DNN using 3 hidden layers (51,51,25). This DNN gave an accuracy of 99%, an F1 score of 0.95 and, a recall of 0.85. This paper shows the DL algorithm can extract patterns to detect more evasive attacks. These patterns have more distinguishable features for the attacks that other ML models can classify.

2.8.2.2 Recurrent Neural Network

Yin et al. [2017] proposed using a recurrent neural network (RNN) for intrusion detection. The models were binary classifiers that were trained using the NSL-KDD dataset. Yin et al. [2017] used an RNN with 80 hidden nodes and a learning rate of 0.1. This model had an accuracy of 83.28% with KDDTest+ test set, and accuracy of 68.55% in the KDDTest21- test set. The FNR rate for KDDTest+ was 27.05%, which is a detriment when using in an IDS. The proposed model does not perform equally well when used for multi-class classification (accuracy on KDDTest+: 77.73%). The model was run after changing the learning parameter to 0.5 and obtained equivalent results to the model used for binary classification (accuracy KDDTest+:81.29%; KDDTest21-:64.27). The accuracy of the RNN model can be improved with better feature selection and input features to the input layer of RNN.

2.8.2.3 Multi-Layer Perceptron

Shenfield et al. [2018] proposed a model using a multi-layered perceptron to detect shellcode in networks packets. The data set they use is the exploit database. The proposed supervised learning model has an input layer of 1000 neurons and an output layer of 2 neurons. The model used 2 hidden layers (30, 30) with a learning rate of 0.01. The model has an accuracy of 80 with FNR and FPR of zero for the test data.

Based on the model proposed by Shenfield et al. [2018], Kanimozhi and Prem Jacob [2019] proposed a multilayer perceptron for signature-based intrusion detection using two hidden layers (9, 4) and alpha parameter of 1e-05. The model was tested on the

CSE-CIC-IDS2018 dataset. The model has an accuracy of 99.975% on the test data for binary classification. The paper shows the effectiveness of MLP in anomaly-based detection. The paper tests the MLP model for binary classification and not multi-class classification. Multi-class classification of intrusions can provide an informed response report.

Xu et al. [2018] proposed a model using MLP and a gated recurrent unit (GRU). The model uses a GRU because it can remember and forget information. The model was trained using the KDDCUPP99 and the NSL-KDD dataset. The GRU model has 128 hidden nodes and the MLP model has 3 hidden layers with 48 nodes. The model has a learning rate of 0.01, an epoch of 20 and, a batch size of 32. This model had an accuracy of 99.24% using the NSL-KDD dataset and 99.84% using the KDDCUPP99 dataset. The detection rate, for NSL-KDD, was 99.31% and for KDDCUPP99 was 99.42%. The detection rate and FPR rate of this model were low. This model is a binary classifier. Xu et al. [2018] mentions the model had low detection rates for U2R and R2L classes. This is due to these classes having fewer records compared to the other classes.

2.8.2.4 Convolutional Neural Network

Khan et al. [2019] proposed using a CNN model that was trained on the KDDCUPP99 dataset. The proposed model uses 3 hidden layers. The hidden layer has a convolutional layer (32-64-128) and a pooling layer. The features were normalized and converted to numeric before training the model. The model has an accuracy of 99.23% when trained using the KDDCUPP99 dataset. The model for multi-class classification shows a learning bias towards DoS attacks which had the most records.

Yong and Bo [2019] proposed a model for intrusion detection using CNN and the Inception structure of the Google team. The proposed model uses the inception model as a layer in the CNN model. The model uses two inception layers, one convolutional layer, 2 sampling layers, one flatten layer, two dropout layers, and three fully-connected layers. The inception model uses 1x1, 3x3 and, 5x5 convolutions and 3x3 maximum samples. The model is trained using the KDDCUPP99 dataset. Similar to Khan et al. proposed model, the features were digitized and normalized. The inception layers help improve the performance and scale of the network. This inception layer was observed to reduce

the dimensionality, weight size, and feature dimensions. The linear activation function 'Relu' was used for all the layers. This model has an overall accuracy of 94.11% , a detection rate of 93.21% and, a False alarm rate of 2.18%. This model has shown high accuracy with a low false alarm rate.

2.8.2.5 Autoencoder

Shone et al. [2018] proposed an unsupervised deep learning model using two non-symmetric deep autoencoders (NADE) stacked over a shallow learning classifier (Random forest). The autoencoder encodes the data to reduce dimensionality and allows for better extraction and learning from the features. NADE uses 3 hidden layers (NADE: 14, 28, 28). The RF classifier is given the encoded patterns made from the two NADE to classify attacks. This model was trained using the KDD CUP 99 and NSL-KDD datasets. This model gives an accuracy of 99.6 %, a precision rate of 99.9% and, a recall rate (97.85%) for KDD CUPP 99. The accuracy dropped when using NSL-KDD to 89.22%. Precision (92.97) and recall (89.22) also dropped. The false alarms rated were also higher in comparison (2.15% to 10.78%). The NSL-KDD trained model metrics are indicative of the actual performance of the model due to several issues in the KDD CUPP 99 dataset [Tavallaee et al., 2009].

Niyaz et al. [2016] proposed the use of self-taught learning (STL) for binary and multiclass classification (5 and 23 classes). The proposed model uses a sparse autoencoder with a SoftMax regression layer. The model was trained using the NSL-KDD dataset. The model gave an accuracy of 88.39% when tested for binary classification. This model was compared with a State-space model (SM) and found overall to be better (accuracy was lower but recall rate and F1 score were higher). Similarly, for 5 class classification accuracy of STL (85.44%) was lower compared to SM (96.56%). The overall performance was of STL (recall: 95.95%; F1 score: 90.4%) was better compared to SM model (recall: 63.73%; F1 score: 76.8%). The use of multiple ML learning techniques is a widely researched topic and the performance of these models should not be judged on accuracy. Using accuracy as the only metric may lead to skewed performances. The evaluation of an intrusion model must be through overall performance.

2.8.2.6 Deep Belief Network

Gao et al. [2015] proposed an intrusion detection model using a Deep Belief Network (DBN). The model was trained using the KDDCUPP99 dataset. The DBN parameters varied are the number of Restricted Boltzmann Machines (RBM) layers and the number of epochs. The RBM is a neural network with a visible and generated hidden layer. The DBN can have many RBM layers but needs at least two layers. The optimal layers were found to be 4 with the number of epochs greater than 150. The model had an accuracy of 93.49%, FPR rate of 0.76% and, TPR rate of 92.33%. The use of unsupervised methods gives less control of the architecture of the detection module. This leads to a model that is biased especially when the dataset has repeated and unbalanced class records.

Alrawashdeh and Purdy [2017] proposed a model that used a DBN model with 2 RBM layers. The model was trained using the KDDCUPP99 dataset. The first RBM layer reduces the input features and is the input for the second RBM layer. This second layer output is given to a logistic regression classifier unit with a soft-max activation function. This model has an accuracy of 95%, a TPR of 92%, a TNR of 2.0%, and, an FNR of 5.7%. The proposed model was improved using the logistic regression model with soft-max classifier along with the model proposed by Gao et al. [2015]. This model uses 4 RBM layers (72,52,40,5 nodes respectively). The input layer uses 122 nodes. This model gives an accuracy (97.9), a TPR of 97.51%, a TNR of 99.48%, an FPR of 0.51% and, an FNR of 2.48%. The use of deep learning to enhance the features and create patterns for improved classification of the classes is popular in AI research.

2.9 NetFlow Traffic Grouping Based Intrusion Detection Approach

The proposed approach of grouping the attack net-flow traffic records and analyzing the obtained record for patterns has grown in popularity after the rise in the use of DL methods. The DL algorithms can process image data more effectively. Vinayakumar et al. [2017] had grouped the data in KDDCup '99' dataset using a filter of 64 on the grouped attacks and pooled to create a tensor to capture temporal patterns. 10% of the KDDCup '99' dataset was used to create the records. The records were not grouped based on the individual attack, rather was grouped as a single entry. This entry will be

passed to the CNN to form a tensor. The tensor was used to train another ML model. The classifier models used were CNN, LSTM, and MLP. The best performing model for binary classification was a single layer CNN with an accuracy of 99.99%. The best performing model for multi-class classification was a CNN-LSTM with an accuracy of 87.8 %. The paper had not grouped the individual attacks into groups, rather formed records using a CNN on all the netflow traffic of the attack on the dataset. The results of this paper show that the grouped traffic record have temporal patterns that can be detected using DL models. The model performance was compared to other benchmark models used with the dataset and did not match the performance of those models. The paper highlights that combining ML models may not always result in performance improvement. The Binary Classifier models' accuracy decrease when combined with other DL models.

Li et al. [2017] had proposed a similar approach using the NSL-KDD dataset. The attack records are grouped based on the attack after preprocessing the dataset. These grouped attacks are converted to a binary vector with 464 dimensions and converted into an 8*8 grayscale image with a padding of 0. The best binary classifier model was a CNN model with an accuracy of 81.84 on the GoogLeNet NSL-KDD Test21 test set. This paper adds to the Vinayakumar et al. [2017] proposed model by implementing a similar approach to the NSL-KDD set. The records are split into binary vectors at intervals of 10 after preprocessing. These segments can have NetFlow traffic from other instances of the same attack. This results in abnormal patterns being formed by the CNN model. The performance metrics being low on the test set can be attributed to this issue.

Potluri et al. [2018] had expanded on the proposed approach of Li et al. [2017]. The attacks were grouped and converted to 464 binary vectors. These binary vectors were converted to 56-pixel greyscale image and padded with zeroes. This image was fed to the CNN model. This approach was tested on the UNSW-NB 15 and NSL-KDD dataset. This approach gave an accuracy of 91.14% for the NSL-KDD dataset and an accuracy of 94.9 on the UNSW-NB 15 dataset. Potluri et al. [2018] expands and improve this approach. The minority classes are not detected and are misclassified as majority classes. This is due to the number of flow entries are not enough to make many records.

Corsini et al. [2021] had proposed a NetFlow grouping method using temporal patterns. The CICIDS2017 dataset is used to implement this approach as it is considered a benchmark IDS dataset. The approach isolates the attack-specific attack classes and examines them for repeated patterns at time intervals. This time interval is then used to group the Netflow attack records. The approach was tested using the FNN-LSTM and LSTM models. The best performing model was the LSTM with an accuracy of 99.8 %. This approach shows promise but is difficult to implement. The detection of temporal patterns of an attack is difficult and the temporal patterns can differ for variations of the same attack. The CICIDS2017 dataset has many variations of the same attack such as DoS that have different temporal patterns. The creation of normal traffic records is difficult because of the variation in temporal patterns as a result of the request type of service. This approach cannot classify minority classes due to the low number of NetFlow records that leads to unnoticeable temporal patterns.

2.10 Critical Analysis

Intrusion detection is an essential security solution to the modern digital world. The rise of devices connected to large networks has increased the demand for effective means of ensuring security and privacy for users. An IDS is a security solution to warn users when they are under threat. The detection of threats improves with the knowledge of different threats, how it is carried, what effects it has on the system or network, and the target of the attack. The analysis of current IDS architecture, implementation, and defects provide an understanding of the challenges and improvements facing IDS technology. The security solutions provided to users can be improved using this knowledge.

Anomaly-based IDS are used for their capability to identify all forms of attacks in theory. The anomaly-based IDS can detect known attacks and unknown attacks. Anomaly-based intrusion detection takes time to detect attacks and this delay can be detrimental in a real-world environment. When making an anomaly-based IDS, the time for data preparation and processing, and model prediction have to be considered. A signature-based IDS is used for its speed and efficiency in detecting known attacks. Signature-based IDS detects attacks using different filtering methods to compare data on a dataset of well-known attack signatures. The delay in detection depends on the filtering method. An anomaly-based IDS performance depends on the used model and dataset. The accuracy

of an anomaly-based IDS can vary, but signature-based IDS can ideally always detect a known attack.

Anomaly-based and signature-based IDS are used together for better and quicker detection of attacks. These systems are called hybrid systems. The network traffic is collected, processed, and analyzed for the packet signature. The extracted packet signature is compared to the signature dataset. This module reduces the delay to alert and implement the countermeasures against known attacks. The anomaly detection unit is a second detection module that will activate if the signature-based intrusion detection module does not trigger. This will result in advantages of both types of IDS. These IDS will require more system resources and have an increased delay in detecting unknown attacks due to the signature-based detection module.

There are many intrusion detection datasets. There are no available datasets that cover all activities in a network or device when an intrusion occurs. Popular datasets used in intrusion detection research are KDDCUPP99 and NSL-KDD. These are used since they adhere to the standard for IDS datasets. These datasets contain many variations of normal behavior profiles which are important for detection. There are different types of recorded attacks. The datasets have traffic and basic features that help in identifying patterns for attacks over the connection type and network activity [Tavallaee et al., 2009]. These datasets have redundant records and classes are imbalanced. The redundant data can be dropped using data managing tools such as pandas. Collecting more data can solve class imbalance but is not always possible. The under-sampling technique removes majority class samples, while over-sampling adds duplicate instances of the minority classes. The under-sampling technique should be avoided as removing instances results in loss of information.

The CISIDS2017 and CSE-CIC-IDS2018 are recent popular datasets in IDS research. The CISIDS2017 dataset is used to train lightweight models. The CSE-CIC-IDS2018 added more entries compared to CISIDS2017 for complex intrusion detection models. These datasets provide the attacker's and victim's IP addresses with is valuable for IDS research. These features are unavailable to examine in the KDDCUPP99 and NSL-KDD dataset and hence ML intrusion model will have to learn without this feature information.

The preprocessing of the dataset is as important as selecting and planning the dataset and ML model. The ML model learns more well-defined patterns after removing unwanted and redundant data. The dimensionality of the dataset reduces when the numeric features are normalized and data with low variance are removed. The selected features should provide information on all the classes in the dataset. These selected features for intrusion detection should not have high correlation values among themselves. Highly correlated features provide similar information and can lead to biased patterns being created by the model [Farahani, 2020].

High-correlation features are can make patterns for classes in datasets with fewer entries. In the KDDCUPP99 dataset, U2R and R2L classes have few entries, so using high correlated features for these classes can make a model that can differentiate these classes. Using high-correlation features results in misclassification of classes due to a biased model. When the trainset is made for the model, the classes should have enough records for the model to detect patterns. IDS datasets have an issue with class imbalance. A solution for class imbalance is to merge similar classes with few records to create a new class. This new class will have more records, hence can detect patterns for the combined classes. Classes with many records can be divided into multiple classes with fewer records. The manipulation of classes in this manner can cause the multi-class classification accuracy to drop.

Intrusion detection using DL algorithms shows promising results. The first step in IDS research is the selection of a dataset. The selected IDS dataset should meet the standards for IDS dataset selection. Popular IDS datasets are KDDCUPP99, NSL-KDD, CISIDS2017, and CIDDS-001. Feature selection can be done to improve the effectiveness of the dataset. Supervised DL techniques are preferred in IDS research as these models allow control over the learning process and output. The learning process can be influenced by adjusting the hyperparameters. Intrusion detection models can be made by cascading ML techniques shows higher results. Shallow ML model's performance can be improved using the features extracted or encoded using DL methods. Thes extracted or encoded data can have reduced dimensionality and more evident patterns for detection.

The patterns created in DL techniques have shown that they can perform binary classification with high accuracy. Another classifier can use these extracted patterns as

features. These patterns have less redundant information. This allows the next classifier to learn from the patterns for evasive relations in features. This results in a more accurate detection model. The effectiveness of combined ML models is shown in [Alrawashdeh and Purdy, 2017, Li and Zhang, 2019, Niyaz et al., 2016, Pawlicki et al., 2020, Xu et al., 2018].

The effectiveness of DL methods for binary classification is shown in many papers such as [Gao et al., 2015, Khan et al., 2019, Tang et al., 2016]. A model used for binary class classification is not as effective for multi-class classification. The cascaded ML models are an effective way to obtain more information from low correlation features in a dataset. The use of two deep learning techniques can provide better pattern generation and recognition. These models requires more time to trained and detect intrusions. The use of shallow ML techniques to classify the patterns made by DL techniques provide a more time-efficient solution. Supervised shallow ML methods such as SVM or RF can classify DL generated patterns [Shone et al., 2018]. Unsupervised methods such as clustering can be effective for multi-class classification. Cascading more ML techniques is redundant and does not increase the performance of the intrusion detection model. Adding more ML methods to the model causes more relations then there are to be inferred from the features leading to misclassification and overfitting of classes.

The grouping of Traffic and net flow records detects time and category-based patterns. The grouping of records in the dataset can be done using communication sessions and protocols. Grouped records can also be made using attack classes, time-slots and, windows or frames. CNN models can be used to compress and pool the traffic data into groups. The grouping of traffic will lead to some traffic records being removed from the performed attack instance and moved to the neighboring group. To prevent this an ideal split condition has to be found but this can be difficult due to the nature of network attacks. The single attack can have many variations with different patterns at different time instances. DL models such as RNN, LSTM, and DBN are known to classify time series sequences but these models cannot effectively detect patterns from fragmented data. Therefore, looking at the whole attack net flow communication is an effective way for ML models to learn the patterns in the grouped data.

Classifier	Method	Parameters	Advantages	Disadvantages
SVM	Classification of classes is performed by using max separation between classes and creating hyperplanes in an n-dimensional space.	The hyper-parameters are penalty parameter 'C', gamma and, kernel algorithm. The SVM performance is affected by the selected kernel algorithm.	• SVM can classify unstructured data. •SVM reduces the dimensionality of data. • SVM experiences less overfitting compared to other models	• SVM is sensitivity and is therefor hard to tune hyper-parameters. • The training and testing time high is due to the use of complex quadratic equations for classification.
KNN	KNN classifies samples by counting the number of class that is close to the sample. If the sample has more class close to it then, using probablity it must be that class.	This model depends on the number of nearest neighbor 'k' and the distance of the sample taken 'p'.	• Training and response time is high. • Easy to implement • Accuracy does not vary when the dataset is scaled.	• Is susceptible to noise and dataset issues such as missing data, large numbers and, high dimensionality. • KNN cannot classify outlier data.

DT	The algorithm is modeled after a tree with branches representing the rules and leaves representing the selected feature. The branches extend until a single class output is obtained. A single DT can give one output. Therefore, multi-class classification creates many decision trees.	Parameters include depth of the tree, number of samples required to conduct split, number of features used for decision tree.	• Does not require much preprocessing of the dataset • Can classify high-dimensional data. •Is easy to visualize and easy to follow the classification process. • Classification can be done even with dataset flaws.	• Tree structure depends on the dataset. The created tree structure can change drastically if a small change is made in the dataset. • More features will result in a larger and complex tree structure.

TABLE 2.3: SHALLOW MACHINE LEARNING METHODS [Gumusbas et al., 2020, Hodo et al., 2017, Xin et al., 2018]

Classifier	Method	Parameters	Advantages	Disadvantages
DNN	DNN Is a feed-forward neural network that can use more than one hidden layer. The parameters are learned from using unlabeled data. The model is tuned by repeating the learning process using labeled data.	The hyperparameters are the epoch rate, number of hidden layers, The number of nodes per layer and activation function.	•The learning process will make patterns even incomplete data.	•Requires more records to train the model. •the hidden layer •Structure of hidden layer is made through trial and error

MLP	MLP is an ANN network with more than 3 layers. The MLP model has an input layer, an output layer, and a hidden layer. The model can use multiple hidden layers which are fully connected.	The hyper-parameters are activation function, number of nodes, learning rate, epoch, batch size and, alpha parameter.	•MLP can be used for classification or regression based on the activation function of the output layer.	•The architecture of the layers in an MLP cannot be known. •MLP overfits the data. •Hard to tune hyper-parameters for MLP.
CNN	CNN stores and learns data using arrays. CNN uses 3 different layers for different this purpose. The convolutional layer learns from features. The Pooling layer samples records. The classification layer classifies the data.	Hyperparameters are activation function, number of hidden layers, number of units per layer, learning parameter, and epoch.	• High accuracy of classification of similar data. •Can use pooling layers to reduce overfitting of model and size of data.	• Has long training times. •Cannot learn and form relations from small datasets (few entries).

RNN	RNN is a neural network that remembers information by feeding the output of a layer as input of the previous or same layer.	Hyperparameters are the number of hidden layers, the number of nodes per layer, activation function, learning parameter, and epoch.	• The configuration of the layers results in the pattern being generated depending on the previous layer.	• Is prone to vanishing gradient (small weights assignment) and exploding gradient (high weight assignment) •Difficult to tune hyperparameters
DBN	DBN is a semi-supervised model that uses RBM models to create patterns. The RBM is a neural network with one visible and one hidden layer. The hidden layer is generated using ML.	Hyperparameters are the number of RBM, number of nodes in the RBM layers, learning rate, and epoch.	•Is used to solve the vanishing gradient problem that is common with neural networks	•Tuning Hyperparameters is difficult •DBN is sensitive to hyperparameter tuning

Autoencoder	Is an unsupervised method that tries to map the output to an input vector.	Hyperparameters are the number of nodes in the middle layer, the number of layers, number of nodes per layer, and loss function.	• Requires less computational resources •Can be used for various tasks like feature compression, classification (versatile)	•Learning process has a high dependence on the dataset, any changes in the dataset will change drastically affect the model. •Autoencoders capture all information in features rather than relations between features

TABLE 2.4: DEEP LEARNING ALGORITHMS Berman et al. [2019], Gumusbas et al. [2020], Hodo et al. [2017], Lazarevic et al. [2005], Xin et al. [2018]

Chapter 3

Requirement Analysis

The chapter defines the functional and non-functional requirements that should be met in this thesis. The evaluation and rudimentary procedure of the requirements will be explained in chapter 4.

3.1 Functional Requirements

The functional requirements are requirements that are the steps and aims the thesis has to achieve. The functional requirements are grouped as mandatory and optional. The mandatory requirement has to be met for the completion of the project and is the priority. The optional requirements provide goals to highlight the primary objectives and will be completed based on available time.

3.1.1 Mandatory Requirements

• Evaluate and select the available IDS datasets where the proposed Netflow traffic grouping method can be implemented.

• Analyse and preprocess the selected dataset.

• Analyse the available traffic features to detect meta-features and create descriptive super features.

• Implement and evaluate the effectiveness of the proposed approach.

• Create subsets of the selected dataset using feature selection algorithms and evaluate them.

• Implement and evaluate the use of deep learning algorithms as feature extractors for shallow learning methods.

3.1.2 Optional Requirements

• Implement and evaluate more deep and shallow learning algorithms for Anomaly-based IDS detection modules.

• Implement and evaluate the use of deep learning algorithms as feature extractors for deep learning methods.

• Evaluate the effectiveness of the proposed approach and models in a real-world environment.

3.2 Non- Functional Requirements

The optional functional requirements will be completed if the time is available. The optional functional requirements for this project are:

• GUI

• Preprocessing and Detection Time

• Open Source

• Version Control

• Platform Compatibility

• Readability

Chapter 4

Methodology

4.1 Evaluation

4.1.1 Functional Requirements Evaluation

4.1.1.1 Mandatory Requirements Evaluation

The pandas python framework will be used to analyze and preprocess the dataset. This framework provides many tools to simplify data manipulation tasks.

• **Selection of Dataset**

Ring et al. [2017], Sharafaldin et al. [2017], Cordero et al. [2021] proposed standards was considered when dataset were evaluated. After evaluating the available IDS dataset, the NSL-KDD, CICIDS2017 and CSE-CIC-IDS2018, and CIDDS-001 and CIDDS-002 datasets were selected. These datasets have detailed documentation and are used widely in IDS research. The dataset will be analyzed and will be selected based on the ease of implementing the proposed approach.

• **Dataset Preprocessing**

The steps for preprocessing the selected dataset are:

• The dataset files are consolidated into a single dataset. The records to be used are selected. If the selected records have missing fields, they will be excluded.

• The dataset is analyzed for issues such as class imbalance and redundant records.

• All the selected features in the dataset are made numerical. The nominal features are label or oneshot encoded .

• The features are normalized to lower the scale.

• These files may have many records which are not required. The extra records are removed, so the classes have a similar number of records. The resultant dataset is divided, forming the test and train sets.

• Feature Analysis and Super-Feature Creation

The IDS datasets will have traffic features and may have descriptive features. The descriptive features in the dataset describe the attack, attack type or, provide information about the entries. Theses features will be removed as they are not meta-features. The dataset's features are analyzed to extract or create super features.

The super features will be evaluated using the feature selection algorithm and performance on the model.

• NetFlow Traffic grouping of Individual Attack Records

A single net-flow record can be considered as a single line of communication between the devices. The aim is to find patterns within the whole communication that can help detect and classify the attacks. The net-flow records are grouped based on individual attacks and fed to ML models.

The approach will be evaluated by comparing the performance of the trained to other models trained using NetFlow grouping.

• ML Models

The project focuses on an ML-based intrusion detection modules. The module is implemented using various ML algorithms. DL classifiers used are DNN, CNN, RNN and, DBN. Shallow classifiers used are DT, SVM, KNN. The ML models will be evaluated on TPR, FPR and, precision rates. The best models will be decided based on the evaluation metrics discussed in Chapter 2.

The model's performance in binary and multi-class classification will be evaluated.

• **Feature Selection**

The feature selection aims to remove redundant and unwanted features from the unified dataset. If feature selection is not performed, the model will take more time to train and detect intrusions. High-correlation and unwanted features will cause the model to learn unintended relations and form patterns from these relations. The feature selection algorithms that will be used in the project are IG and CFS.

The models will be first trained on all the meta-features. Different models will be trained using the feature subsets and compared to models trained on all the features. The models trained with feature subsets are evaluated on the models trained with all the features.

• **Combination of deep and shallow learning methods**

The DL techniques to be used as feature extractors are CNN, RNN. These feature extractor models output is tested on shallow learning models. The shallow learning classifiers used are DT, KNN, SVM.

The Combined models will be evaluated with the performance of the models using single ML methods.

4.1.1.2 Optional Requirements Evaluation

• **Optional ML methods**

Optional DL classifiers to be used are MLP, LSTM and, Autoencoders, and Shallow classifiers to be used is RF.

• **Combination of deep and deep learning methods**

The DL techniques to be used as feature extractors are CNN, RNN. These feature extractor models output is tested on deep learning models. The DL classifiers used are RNN, LSTM and, MLP.

• **Evaluation of Models on Real Networks**

The documentation for datasets provides the tools and scripts used to collect the network traffic. Network traffic can be captured with tools such as Wireshark or pyshark python package. The provided scripts can be run on the collected traffic data and converted

to the dataset format. These records will be fed to the ML pipeline and given for classification to the ML models.

4.1.2 Optional Requirements

• GUI

A GUI to view the results of the thesis will be made. The GUI will be evaluated with the number of bugs and compliance with Software development standards

• Preprocessing and Detection Time

The training and detection time of the created models will be monitored and, steps will be taken to reduce this using more efficient code. The proposed approach and models are hypothesized to take more time to complete due to the additional preprocessing and bulkiness of the records.

• Open Source

The project will be implemented in the open-source coding language Python. Paid software or tools will not be used in the project. The Jupiter notebooks and code will be shared on the GitHub platform under the GNU licence [GNU].

• Version Control

The source code used for the implementation of the project will be explained through comments with proper indentation. This readability will be evaluated using the number of bugs and clarity of the code.

• Platform Compatibility

The project is implemented on python version 3.7.4. The python kernel is a cross-platform compatible kernel. The Python coding language can be downloaded and used in Linux, Mac, and Windows. The source code can be run by downloading the python kernel and the associated python packages to the system.

• Readability

The standard best practices and approaches when be followed when writing the source code. Appropriate Comments will be provided when required to explain the source code. Proper indentation of the source code to improve readability will be used.

Chapter 5

Professional, Legal, Ethical and Social Issues

5.1 Professional Issues

1. This project strives to uphold the IEEE CS/ACM Code of Ethics and Professional Practice [Gotterbarn et al., 1997].

2. The code that is developed, designed, and tested will follow the software engineering best practices as prescribed in [Jones, 2010].

3. This project ensures that all toolkits, software, datasets, and libraries used for the development of the project will have appropriate licenses that allow them to be used. Credits and references are provided wherever needed in keeping with the BCS Code of Conduct regarding third parties.

4. All books, articles, and references in this paper will be referenced and cited properly. Any code snippets obtained from outside sources used in the project will be acknowledged. The source code of this project will be open-sourced abided by the policies and published under a GNU General Public License [Free Software Foundation, 2007] which permits to use, modification, and distribution of the code.

5.2 Legal Issues

1. The project does not violate any data laws that are applicable in the university and country.

2. Other tools used in this project will be open source and will be used following their license and terms of use.

5.3 Ethical Issues

1. This project does not use human subjects for testing purposes.

2. No confidential or privately-owned data is being used in the project. The datasets used are publicly available for use. The data has been verified by the creators and is intended for research purposes.

3. This project is related to intrusion detection, analysis of network traffic can lead to private data being found. The models may be, if time permits, will be testing in a real-world environment. This will use traffic analysis in a simulated environment. If any private data is found during this process it will not be used for harm and will be kept confidential.

5.4 Dataset

The CIDDS-001 and CIDDS-002 are publicly available datasets for intrusion detection. The dataset license can be obtained for IDS research by referencing the author's research paper [Ring et al., 2017, **?**].

The research papers are:

Ring, M., Wunderlich, S., Grüdl, D., Landes, D. and Hotho, A., 2017, June. Flow-based benchmark data sets for intrusion detection. In Proceedings of the 16th European Conference on Cyber Warfare and Security. ACPI (pp. 361-369).

Ring, M., Wunderlich, S., Grüdl, D., Landes, D. and Hotho, A., 2017. Creation of flow-based data sets for intrusion detection. Journal of Information Warfare, 16(4), pp.41-54.

5.5 Social

1. The data used is obtained from publicly available datasets, any data in these have been checked and is made public after verification. Therefore, no confidential data about people available in the project.

2. The purpose of this project is for research. The stakeholders do not condone or intend this project to be used for any harm to others. If used for harmful or illegal activities, the perpetrator is solely responsible.

5.6 Safety

1. There is no equipment or wearables that are being used for this project. Any testing will be done in a digitally closed and simulated environment without any human test subjects.

2. Meetings with stakeholders were done online due to the current global pandemic.

Chapter 6

Implementation

6.1 Experimental Setup

The project is implemented using the Python programming language. All data analysis, preprocessing, and model building was done with the python programming language. The pandas and sklearn python packages were used for data preprocessing and analysis. Sklearn and TensorFlow python packages were used to design and build the ML models.The Jupiter Notebook and Visual studio Code IDE were used to write and test the source code. The GUI is implemented using the Tkinter python package. The graphs and plots are made using the Seaborn, Matplotlib, and Tensorflow python packages.

The ML models were run on a system with the specifications:

Processor: Intel(R) Core(TM) i7-8565U CPU @ 1.80GHz 1.99 GHz

RAM: 32.0 GB

Operating System: Windows 10 Pro

6.2 Evaluation and Selection of Dataset

Ring et al. [2017], Sharafaldin et al. [2017], Cordero et al. [2021] proposed standards was considered when dataset were evaluated. After evaluating the dataset, the NSL-KDD, CICIDS2017 and CSE-CIC-IDS2018, and CIDDS-001 and CIDDS-002 datasets

were selected. The NSL-KDD dataset is popular for IDS research, but it was not selected because it is outdated.

The CICIDS2017 and CSE-CIC-IDS2018 are recently created IDS datasets. The datasets have 13 classes, which are 12 attacks and begin traffic. Both the datasets have class imbalances. A solution to this is to join similar minority classes as a single class [Panigrahi and Borah, 2018]. The dataset documentation provides details of the timings of the performed attack and the attacker's and victims' IP addresses. The datasets have a large number of attack records and 80 features. The dataset was not selected because even though the documentation Sharafaldin et al. [2018b] details the time slots for the performed attacks, it does not document the number and timings of individual attacks. The implementation of this approach is difficult in this dataset.

The CIDDS-001 and CIDDS-002 are recent IDS datasets that strictly follow the standards mentioned by Ring et al. [2017]. The dataset has records that are run in a closed and simulated, and real-world environment. The documentation provides details on the IP address in the subnets in the simulated environment. The external environment has complex network traffic with IP addresses that are not a part of the simulated network. These external IP addresses are anonymized for user privacy. The attackID field in the datasets is useful to group the records related to an individual attack. The CIDDS-001 dataset has 14 features and 90 recorded attacks. The CIDDS-002 dataset has 14 features and 43 recorded port-scan attacks. The documentation does not provide details of the attacker, such as their IP addresses or the subnet. These datasets were selected because it is simple to implement the proposed approach. The CIDDS-002 dataset is not selected because the dataset only has entries for variations of ping scan attacks. The inclusion of all the CIDDS-002 dataset classes will lower the classification accuracy. If the classes in the CIDDS-002 dataset are merged to portScan class present in the CIDDS-001 dataset, it will lead to class imbalance. Hence the CIDDS-001 dataset is selected for the implementation of the project.

6.3 Preprocessing Of Dataset

The CIDDS-001 dataset comes with the configuration, log and, net-flow files in CSV format. There is a total of 8 net-flow traffic CSV files. The project uses 5 files where

two are from the Openstack and 3 are from the External environment. The normal traffic records from the 3 of the files are removed before combining to form a single dataset.

The normal traffic records in internal week1 and external week4 files were used to have a mix of simulated and real-world normal traffic. The attack records from all files were taken. The attackID field in the external files was incremented by 70, which is the number of attacks conducted in the Openstack environment, since both environment's files attackID fields start at 1. Any string columns that can be converted to numeric are converted.

The unified dataset is analyzed for any missing or null values in the records. Missing values cannot be dropped and appropriate values would need to be substituted. These records cannot be dropped as this will lead to loss of information when the records are grouped. The analysis shows no missing fields in the records. After this, the super feature creation, feature selection and the proposed process can be performed.

After super feature creation, the meta-features are normalized using min max normalization. The dataset is grouped based on the attackID field. The normal traffic net-flow records are grouped as a single entry and processed to form normal profiles. The descriptive and non-meta features are removed. The dataset has 34 features, not including the class field attackType.

6.4 Extraction of Super features

The meta-features in the dataset are analyzed for any features the can be extracted or created. The analysis and research show the Source and destination IP and the Flags fields can be used to extract super features.

The Flags field has two types of values, one type of value is a string that shows which TCP flags are activated and the other value is a hexadecimal value that can be converted to binary and denotes the activate TCP flags. This field is converted to a sparse matrix with individual flags as features. 9 additional features are added to the dataset with the values 1 for raised and 0 for not raised. This method will produce a dataset that has reduced dimensionality compared to if the original Flag field is oneshot encoded. If

the Flag field is label encoded, unwanted numeric relations will be made between the grouped records when used to train ML models.

The Source and Destination IP address features are hard to encode due to their nature. These fields provide valuable information for attacks such as DoS and portScan but cannot be directly learned by ML models as they are not quantifiable [Sharafaldin et al., 2018a]. Shao [2019] had conducted a study on ways to encode IP addresses. The paper compared the performance of oneshot label encoding, binary IP splitting, and split IP. It was found that splitting the IP address to their respective network and host sections gave better model performance compared to other IP encoding methods.

The CIDDS-001 dataset has anonymized the IP addresses that are not a part of the OpenStack environment and the attacker's IP address. The IP addresses fields have additional values that are EXT_server, OPENSTACK_NET, and DNS. The value OPENSTACK_NET represents the public address of the Openstack server, EXT_server value represents the IP address from the external server, and DNS value represents the shared DNS IP address of the servers [Ring et al., 2017]. Since the networks are known, they are added as individual features. Adding these values as features lets the model learn this relation. These features have values 1 and 0. The other 4 IP features values are set as 0 when any of these 3 features are set as 1. The attackers and anonymized IP addresses are replaced with the values 0.0.0.0 in the fields. IP addresses not a part of the network can be replaced with this value to create a relation. This approach decreases the number of IP address values in the IP fields. This lowers the information gained from these features and leads to better feature selection.

This approach of encoding IP addresses provides more descriptive meta-features. Oneshot encoding the IP address leads to an increase in dimensionality based on the number of IP address values. This approach splits the IP field into 7 features. The IP addresses that are not a part of the subnet or belong to another known subnet are given as features. IP addresses that are not a part of the network are not given a value. The normal traffic in the CIDDS-001 external files shows normal behaviour profiles using the anonymized IP addresses. This results in the model not learning any unwanted relations because the anonymized attacker fields are assumed to be outside the network. After super feature creation, 23 additional numeric meta-features were added.

Dataset Features (Total: 34)					
IP		Net-Flow	TCP Flags		Protocol
sourceIP_feature 1	destIP_feature 1	Duration	N	P	GRE
sourceIP_feature 2	destIP_feature 2	Src Pt	C	R	ICMP
sourceIP_feature 3	destIP_feature 3	Dst Pt	E	S	IGMP
sourceIP_feature 4	destIP_feature 4	Packets	U	F	TCP
sourceEXT_SERVER	destOPENSTACK_NET	Bytes	A		UDP
sourceOPENSTACK_NET	destEXT_SERVER	Tos			
sourceDNS	destDNS				

TABLE 6.1: Dataset final features

6.5 Feature Selection

There are 34 meta-features in the dataset after feature extraction and oneshot encoding the nominal value. Some features provide more information compared to others. These features can be selected manually or using feature selection algorithms. This project will create feature subsets using IG and CFS.

The 34 features in the dataset include 14 split IP features. The first feature subset will remove the Source and Destination IP features. The first subset is created to evaluate the importance and efficiency of the split IP feature extraction.

The second and third feature subsets are created using the CFS feature selection algorithm. The second feature subset is created by applying the algorithm to all the features. The third feature subset is created by applying the algorithm to the first feature set. The third feature subset is used to evaluate highly correlated features that do not include the split IP features.

Similar to the second and third feature subsets, the fourth and fifth subsets are created using the IG feature selection algorithm.

The first subset had 20 meta-features. The second feature subset has 7 correlated features that were selected when the CFS algorithm was run on the whole dataset. The third feature subset has 6 correlated features that were selected when the CFS algorithm was run on the first feature subset. An open-source python implementation of CFS algorithm was used in this project [ZixiaoShen, 2019].

The fourth feature subset has 11 features that were selected using the IG algorithm when it was run of the whole dataset.The fifth feature subset has 9 features that were selected

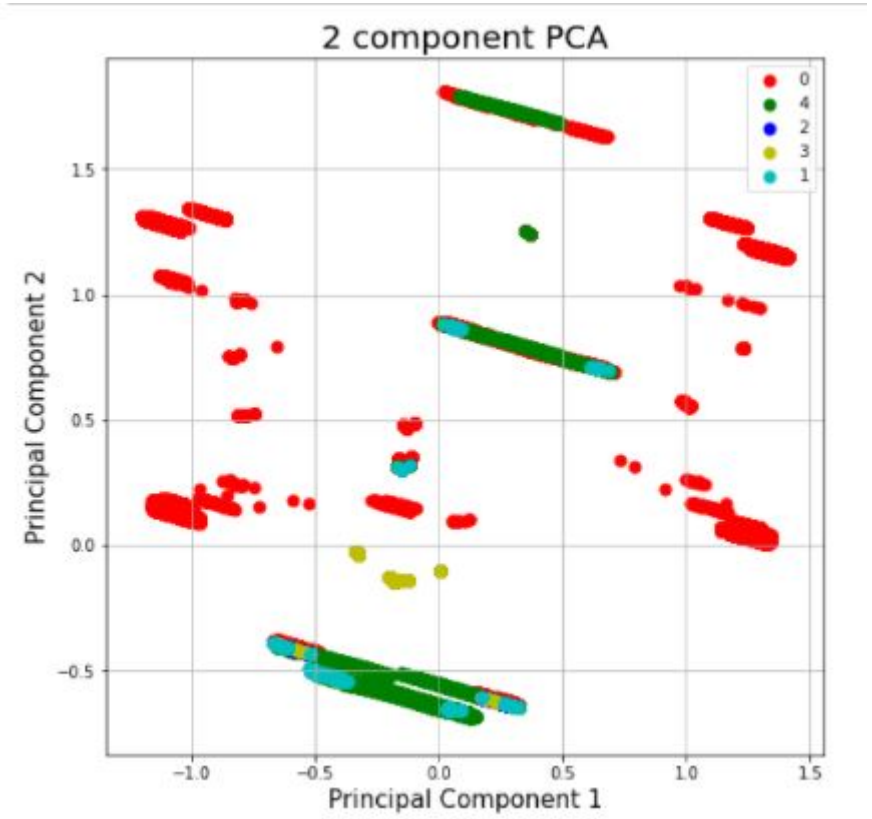


FIGURE 6.1: PCA Analysis of CIDDS-001 Dataset

CFS Features Subsets	
Feature subset -2	Feature subset- 3
F	S
S	R
sourceIP_feature 4	Packets
destIP_feature 4	Tos
Duration	Duration
sourceIP_feature 1	P
Packets	

TABLE 6.2: Features Selected Using CFS Algorithm

using the IG algorithm when it was run of the first feature subset. An open-source python implementation of IG algorithm was used in this project [IG-, 2021].

6.5.1 Implementation of Grouping Attacks and Normal Traffic

Most IDS research classifies each net-flow record by finding relations between the net-flow packet features. This project considers each net-flow record as a feature, with

IG Feature Subsets			
Feature Subset - 4		Feature Subset - 5	
Bytes	Packets	Bytes	TCP
sourceIP_feature 4	S	Src Pt	UDP
Src Pt	F	Dst Pt	ICMP
Dst Pt	sourceIP_feature 3	Duration	
destIP_feature 4	destIP_feature 3	Packets	
Duration		Tos	

TABLE 6.3: Features Selected Using IG Algorithm

the whole record having multiple net-flow records. This approach classifies the attacks based on the patterns found in the communication between the devices. The CIDDS-001 dataset labels the conducted attacks using the attackID field. The records are grouped using this field and regarded as a single record.

The next challenge was the formation of normal behavior records. The timestamp field was converted to associated UNIX time in milliseconds from a string value. The normal records were grouped and a single record was formed. The single record was sliced into groups in the range of 3 hours (10800000 Unix time), then records with no entries were removed. This range was selected as it creates records with a varied number of normal net-flow entries. This results in varied normal profiles for the ML models. The DoS records were removed due to hardware constraints.

The attack and normal records do not have the same number of entries and need to be padded. Popular padding techniques for ML are post-padding and pre-padding. Lopez-del Rio et al. [2020] surveyed the various padding techniques to provide a benchmark. The padding techniques were tested using protein functional prediction with raw amino acid sequences using DL algorithms. Lopez et. al found zoom, strf, mid and, post padding techniques provide better performance with the DL model. The performance of zoom and strf padding was overall superior compared to post and pre padding. Dwarampudi and Reddy [2019] survey the use of post and pre padding on LSTM and CNN. The results show that pre-padding drastically improves the performance of LSTM, while the performance of CNN only improved slightly. The paper mentions using pre-padding when combining multiple types of neural networks are more effective. The paper does not provide evidence to support this claim. In this project, the records will be post-padded with a zero matrix with the number of features. The other padding methods

did not show large performance improvement in the ML models [Lopez-del Rio et al., 2020]. After padding, each record had 19800 net-flow entries.

6.6 Model Creation and Evaluation

Shallow learning models are designed using the sklearn python package because they provide APIs to design, save and load shallow learning models. The deep learning models are designed using the TensorFlow python package. This package allows for modeling and joining of the deep learning layers and can be used to build custom DL models.

The DBN model was not available in the TensorFlow package and hence had to be custom-made. The open-source DBN implementation using TensorFlow was used for this project [albertbup, 2017]. The source code had to be changed slightly to be compatible with the current TensorFlow version. The MLP model was designed using sklearn package.

The shallow learning models hyperparmeters were tuned using a Bayesian Optimizer. The Bayesian optimizer was initialized with the classifier and possible ranges of values for each hyperparameter. This automates the process of hyperparameter tuning for the models. The best model was decided based on log-loss and accuracy of the model. The deep learning method's hyperparameters were manually tuned. Using tuning techniques such as Grid Search or Bayesian optimizer increase the search time due to the number of hyperparameters that can be tuned.

In addition to the Jupiter python notebooks where the ML pipeline was executed, a GUI made using Tkinter will be created to view the results of the models and tune the train-test dataset split to compare the performance of the models with the dataset.

6.7 Summary of Preprocessing and Changes to the Dataset

The summary of the changes and assumptions made to the CIDDS-001 are:

• The anonymized public IP addresses in the dataset are replaced with the IP address 0.0.0.0.

	Normal	Brute Force	DoS	Ping Scan	Port Scan
CIDDS_001	118	26	18	16	32
CIDDS-002	111	0	0		43

TABLE 6.4: Records in the CIDDS-001 and CIDDS-002

• The anonymized attacker IP addresses in the dataset are replaced with the IP address 0.0.0.0.

• The Source IP and destination IP address are split into their network and host segments. These segments are added as features. There are 8 IP features added to the dataset.

• The anonymized known networks and DNS IP addresses are added as separate features. There are 6 features added to the dataset.

• The TCP feature for the TCP communication protocol is split into individual TCP flag features. There are 9 TCP features added to the dataset. • The Protocol feature is oneshot encoded. This results in 5 features being added to the dataset.

• The Timestamp feature is converted to its UNIX time value in milliseconds.

• The attack and normal traffic records are grouped. The normal traffic record is split into smaller records in the range of 3 hours. Normal traffic records with no entries are removed.

• Any records with more than 20000 entries are removed. This results in all DoS records being removed.

• The records are post-padded, and the features are normalized. The descriptive features are removed and the attackType feature is label encoded. The attackType feature will be used as the class fo the ML models.

• Feature selection and ML model depended processing is done. The records are converted to arrays. The dataset is split into a train and test set using a test split of 20%.

Chapter 7

Evaluation And Comparison

7.1 Result Analysis

In this chapter, we will compare and analyze the results obtained using the proposed traffic grouping method. The results for the designed models are available in A.

Splitting the Source and IP address provided more features for the model. During feature selection, the IG and CFS algorithm had selected the host features instead of the network features. The IG algorithm selected these features as these have the most number of distinct values compared to the other features. The CFS algorithm picked these features by measuring the correlation between the feature and the class. The feature selection algorithm shows the host IP features provide more information and are correlated to detecting attacks.

In the Source and Destination IP address features, the IP addresses denoting the known networks and services such as the DNS were removed and added as features. These features, except for the DNS feature, were redundant in the dataset. The DNS IP address provided more information gain compared to the other none network features. This super feature can be used to represent services for the ML model to learn. The known network features can be removed as these are redundant and represented using the split network IP features.

The TCP protocol flags were split into nine TCP flags. These features added more dimensionality to the dataset but provided more descriptive features. The TCP flags

that provided the most information gain are S, F, R, and P TCP flags. This approach to encoding the TCP flags is much more effective than oneshot encoding the TCP flag values provided in the dataset. This approach increases the dimensionality of the dataset slightly compared to oneshot encoding and does not cause the illusion of a more descriptive feature when label encoded. The TCP flags are used to denote a particular state of the connection or provide additional information regarding the connection. Therefore, every TCP flag does not provide the information for the model.

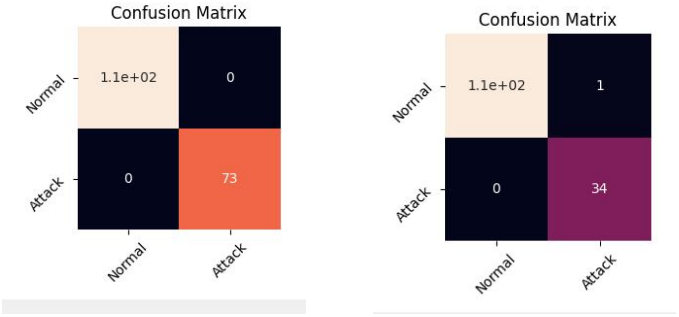
Due to hardware restrictions, the DoS attacks were removed from the dataset. The Traffic grouping approach showed great results when tested on the test and CIDDS-002 dataset. The models were first trained on all the features. The shallow learning models had high-performance metrics for the test set while performing decently well in the CIDDS-002 set. The shallow learning models showed better performance in binary classification compared to multi-class. The precision and recall rates for SL models were low, whereas the FPR rate was high. The SL models were biased to normal records. The SL models perform well in binary classification in the CIDDS-002 dataset with high TPR and precision. The SL models were unable to provide the same performance in multi-class classification. This is a result of the class imbalance in the dataset and the models being underfitted for each attack class. The best performing SL methods are SVM and RF. When the SL models were tested on the created feature sets, the performance did not improve as much. For feature sets 2 and 3, with the least amount of features, the models performed worse than the other feature sets. These models feature sets 2 and 3 trained showed high performance metric values for the test set but performed worse on the CIDDS-002 set. As the number of features has decreased the performance of KNN and DT models reduced whereas the performance of the SVM and RF models improved. The removal of IP features also lowered the performance of the SL models. The SL models were detecting patterns using the IP address values and hence are not reliable when tested on other networks with different IP addresses. The KNN and SVM models performed better with the feature set 1 and all features. The SVM performed better for the feature set 1 and 4, while RF performed better for the feature set 5 and all features. In general, the SL models performed better with all features because the number of features allowing for better learning and classification.

The DL models performed better in both binary and multi-class classification compared to SL models. The DL models trained on all the features in the dataset performed has

a better performance compared to SL models. The DL models have better performance metrics compared to SL models. The TPR rate and precision rates are lower for DL models. When the DL model was trained using all the features, the performance of the models was comparable to SL models. When the model was trained using feature set 1, all the used DL models except MLP showed better performance on the CIDDS-002 dataset compared to CIDDS-002. The MLP model was an outlier and the hyper-parameters can be tuned to give better performance. The models trained using the other feature sets showed high performance metrics as well. The CNN models performed was worse compared to the other models. The CNN models performed worse when the features were reduced. The CNN pooling and convolution layers decreased the dimensions of the records and were better used as feature extractors. The LSTM models performed better than the RNN model, these models are similar in working but the architecture of the LSTM remembers data for a longer time compared to RNN. The RNN model forgets data after a few epochs unless they are repeated frequently, the RNN performance being comparable to LSTM is a result of the dataset's low number of records. The RNN model performed worse compared to LSTM for feature sets 1, 3, and 5 because the model forgets the IP features due to their frequency of occurrence. Similarly, the DBN and DNN models have performed better with the feature sets without IP features whereas MLP performed better for features sets with IP features. The CNN models performed better for feature sets 1 and all features. The performance of the CNN improved with more features similar to SL models. The DNN models performed better for feature sets 3, 5 and, all features. The RNN models performed better for feature sets 3 and 5. The RNN models perform better when trained on the feature set without IP features. The RNN models performed better for feature sets 3 and 5. The LSTM models performed better for feature sets 1, 3, and all features. The MLP models performed better for feature sets 3 and 5.The DBN models performed better for feature sets. The Deep learning models performed better with more layers and nodes, except for DBN. The LSTM, RNN and DBN models were very sensitive to hyperparamter tuning.

The DL models were less reliant on the IP features compared to SL models. This was observed with the consistent performance of the DL models on the different feature sets.The deep learning models performed much better compared to shallow learning models. The deep learning models were better at obtaining patterns from smaller feature sets. Models such as LSTM, DNN, and DBN were able to provide a similar performance

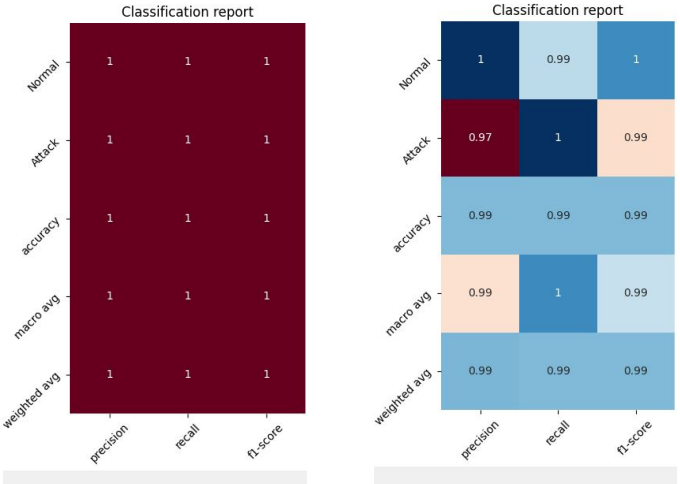
compared to all the models using all the features. The LSTM, DBN and, RNN deep learning model's performance was highly dependent on the hyper-parameters used. Due to exploding gradient, the LSTM, RNN and, DBN models showed infinite log loss and classified the records as a single class. This issue was overcome by using the rmsprop optimizer and, low learning and dropout rates. The CNN models are better used to compress the records and extract patterns to classify.



(A) CIDDS-001 (B) CIDDS-002

FIGURE 7.1: Confusion matrix of DBN - Feature Set -3

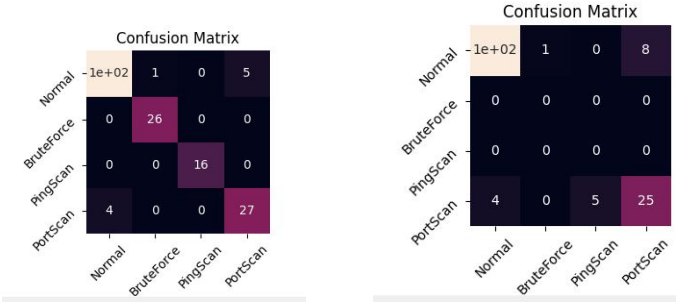
The use of DL models as feature extractors was tested on SL and DL algorithms. CNN, LSTM, and RNN were used as feature extractors. The extracted features are fed to SL and DL models. The use of DL feature extractors did not improve the precision of the models but improved the TPR and FPR rates. This is due to the few attack records per class. The improvement in TPR and FPR rates is desirable for an IDS module. The precision rates are comparable to the single ML models. The low precision



(A) CIDDS-001 (B) CIDDS-002

FIGURE 7.2: Confusion matrix of DBN - Feature Set -3

rate in ML models can lead to the wrong classification of the classes. These combined models improve the precision of binary classification. This analysis shows that with more records, these models can provide better results. The low number of records lowers the multi-class classification rate due to learning from the same patterns from the dame records. The SL models that perform better with the feature extracted are SVM and RF. Using the RNN as a feature extractor lowered the performances of the ML models. The LSTM feature extractor performed much better compared to the RNN. This is a result of the difference in the architecture of the models. When the extracted features were tested on the DL models, the performance improvement was similar to SL models. The improvement in TPR and FPR rates are similar to SL models with a few outliers. The precision rates have decreased compared to single DL models. The lowered performance is due to the over-analysis of the features.

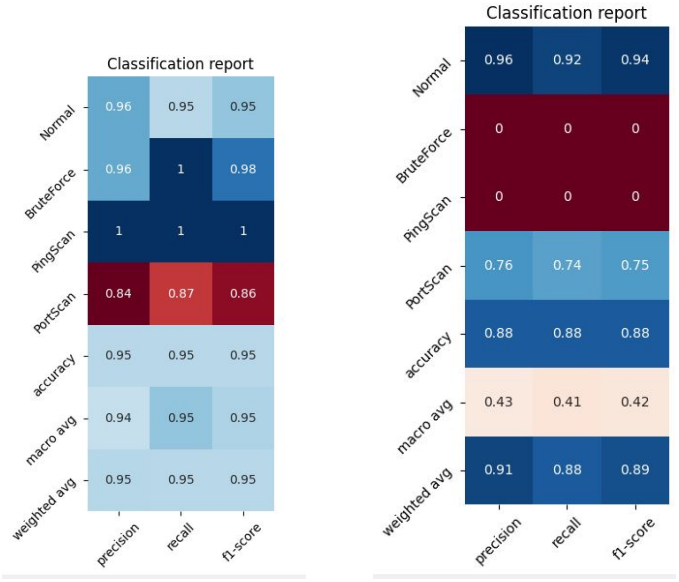


(A) CIDDS-001 (B) CIDDS-002

FIGURE 7.3: Confusion matrix of CNN-Decision Tree - Feature Set -5

The DL models showed better performance for binary classification compared to multi-class classification. Analyzing the ML model's confusion matrix shows that the dataset needs more attack records for ping-scan and brute-force attacks. Most models were able to detect the port-scan attacks. This correction can further improve the performance of the models.

The feature sets were run on all the models. The feature sets with IP addresses performed better with shallow learning models. The SL models made patterns with the IP address features and hence performed worse when tested on the CIDDS-001. All the feature sets performed well for the DL model and the performance shows that DL models were able to detect intrusions without the IP address features. The feature sets with a low number of features also performed well. The feature sets without IP features provide better performance when tested on the CIDDS-002 dataset. The selection helped improve the



(A) CIDDS-001 (B) CIDDS-002

FIGURE 7.4: Confusion matrix of CNN-Decision Tree - Feature Set -5

preprocessing time but the best performing feature set is feature set 1. The feature set 2 and 3 performances were lower compared to the other feature sets in SL and DL-SL models. The feature set -1 had consistent performance on the CIDDS-001 and CIDDS-002 dataset. This shows that the IP features were not required for the ML model to detect and classify intrusions.

7.2 Comparative Analysis

	SVM	RNN	LSTM-SVM	CNN-RNN
Feature-Set	Subset-3	Subset-5	Subset-1	Subset-1
Accuracy	98.9	98.9	99.5	99.5
Precision	98.9	98.3	99.8	99.8
Recall	97.6	98.2	98.4	98.4
TPR	98.9	98.9	99.5	99.8
FPR	0.4	0.4	0.2	0.2
F1-Score	98.2	98.2	99.1	99.1

TABLE 7.1: Best Models (CIDDS-001 - Multi-class)

The implemented approach can be evaluated by comparing to other equivalent implementations. The performance of these similar approaches [Corsini et al., 2021, Li et al., 2017, Potluri et al., 2018, Vinayakumar et al., 2017] can be used to evaluate the performance of our model. These models have been discussed in chapter 2. Table 7.1 and 7.2

	RF	LSTM	CNN-SVM	CNN-MLP
Feature-Set	Subset-1	ALL	Subset-1	Subset-1
Accuracy	99.5	98.9	100	99.5
Precision	99.3	98.6	100	99.3
Recall	99.5	98.9	100	99.5
TPR	99.5	98.9	100	99.5
FPR	0.6	1.1	0	0.6
F1-Score	99.4	98.9	100	99.5

TABLE 7.2: Best Models (CIDDS-001 - Binary)

	CNN	CNN	LSTM	CNN
Classification	Multi	Multi	Multi	Binary
Reference	2018	2017	2021	2017
Accuracy	91.14	81.57	99.78	99.9
Precision	97.82	81.81	99.43	99.9
Recall	93.74	99.63	99.9	99.9
F1-Score	93.06	89.85	99.67	99.9

TABLE 7.3: Performance Comparison of Other Net-Flow Grouping Approaches

shows the performance metrics of our best models. Table 7.3 shows the best-performing model implemented using their proposed approach.

The different models were analyzed using the F1 score, precision, and recall. The implemented approach shows better results compared to compared approaches. The grouping of complete communication for an attack entry results in the patterns being detected by ML models compared to grouping by time and window. The multi-class classification of the model can be improved further with more attack records.

7.3 Thesis Evaluation

As discussed in chapter 4, the TNR, FPR, and precision rates were performance criteria that were used to evaluate the models. The effectiveness of the proposed methods was evaluated by comparing them with other similar approaches. The Feature sets were evaluated on the dataset with all the features. Similarly, the use of deep learning methods as feature extractors was evaluated. Thus, we can conclude that our method has succeeded in meeting all mandatory functional requirements mentioned in chapter 3.

We have met two optional functional requirements. The analysis of the best model in a real-world environment was not completed due to time constraints. Instead, the models were tested on a similar dataset (CIDDS-002) [Ring et al., 2017] to evaluate the effectiveness of the models in detecting different intrusions. The results are added in the appendix A. The results are compared with the performance of the models in the CIDDS-001 dataset.

The best models that were found in our thesis are shown in table 7.1 and 7.2. These models were decided based on the evaluation metrics discussed in 2.

There optional requirements for the thesis are:

- GUI

As per the non-functional requirements in chapter 3, a GUI is set up to view the results of the thesis, shown in . The user can select the feature set, classification type and the dataset to test the models. The DBN model can be tested on another GUI becasues of a TensorFlow version issue.

- Preprocessing and Detection Time

The preprocessing and grouping approach takes around 13-20 minutes for the dataset. Preprocessing a single record will take around 38-50 seconds. The preproessing time is large due to the changes that are made in the dataset. The Feature subsets require less time compared to all the features. The speed of preprocessing was resuced by using effective code and python framework numpy and pandas for preprocessing operations.

The models takes a detection time of 3-4 seconds after preprocessing.

- Open Source

The source code is the written in an open-source python language. Open source development platform used was Jupiter Notebook and VScode. The source code and models are available in the GitHub repository:

https://github.com/Suryavinayakjyothish/F21MP-Dissertation

- Version Control

All versions of the thesis was maintained using the git software.

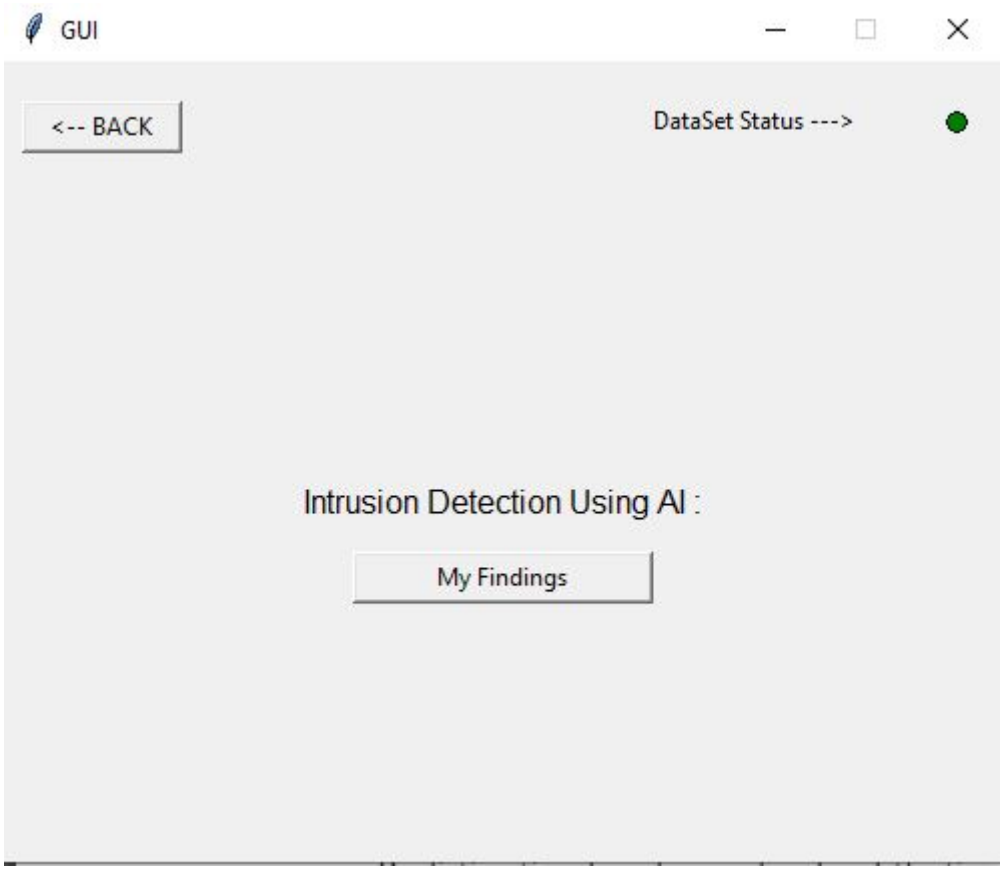


FIGURE 7.5: GUI Start Screen

• Platform Compatibility

The development was done using python programming language which is platform independent.

• Readability

The source code was written with proper indentation and comments.

Chapter 8

Conclusion and Future work

8.1 Conclusion

The first question was the effectiveness of the proposed Netflow grouping of individual attacks. The approach was used to train the models and then compared to other models using similar Netflow grouping approaches in chapter 7.

The second question was the effectiveness of SL and DL methods in intrusion detection. The SL models performed worse compared to DL models. The DL models were able to detect and classify intrusions more consistently when tested on the CIDDS-002 dataset. The SL model's performance improved well with IP features on the test set but it did not translate to the CIDDS-002 dataset. The IP features were used to make unwanted patterns by SL methods. The DL models performed well on the CIDDS-002 dataset with all feature subsets. The DBN and LSTM model was the best performing models for intrusion detection.

The third question is the effectiveness of DL algorithms as feature extractors. The DL methods used to extract the features did not improve the precision and F1 score parameter for SL methods. The TPR and FPR rates improved. The SL models binary classification capabilities improved using the DL extracted features whereas there was only a slight improvement in the multi-class classification. A dataset with more attack records can improve the multi-class classification. The CNN extracted features improved the performance of LSTM, RNN, and MLP models. CNN algorithm provides a way to compress and extract features from large records, such as the records used in this project.

The fourth question was the importance of IP addresses in ML-based intrusion detection models. The IP addresses were split into their network and host segments. Using the feature selection methods shows that the most information gain is in the host segments. The 2 network features provide information when used with all the features provides higher performance compared to the feature subsets. The network features can be used along with DL methods for a more effective model compared to SL models.

The fifth question was the importance of the TCP flags as a feature in ML-based intrusion detection. The splitting of TCP flags reduced the dimensional of the dataset. The feature selection algorithm showed that not all TCP flags are relevant to the ML model. The flags selected using the feature selection algorithm are S, F, R, and P TCP flags.

The sixth question is what are the important Net-Flow communication features. The feature selection algorithms help determine that the net-flow features that have more information gain and are correlated. The features that were selected by feature selection algorithms were Bytes, Src Pt, Dst Pt, Duration, packets, Tos and prototype.

The last question is the feasibility and effectiveness of the Traffic grouping approach. The models were not tested in a real-world but were tested using the CIDDS-002 dataset. This dataset is similar to the CIDDS-001 dataset and can be used to evaluate the performance of the model to detect port scan attacks. These models were able to detect intrusions well but were not able to perform multi-class classification well. This a because the CIDDS-001 dataset has only 92 instances of attacks. This method is applicable in the real world but is hard to implement. The attack traffic needs to accumulate to form a record. one concern when using this approach is the number of traffic flows required to create a record which the model will be able to detect. The preprocessing time is for a single record is 38-50 seconds depending on the number of traffic records. This approach also requires larger hardware requirements from the system. The project had used attack records with traffic flow entries less than 20000, yet faced memory issues when training the models. The feature subsets performed better and improved preprocessing time. The use of a smaller subset of descriptive features can help alleviate the memory requirements. The use of an ML method to decrease the dimensionality or encode the records can improve the effectiveness of the trained models.

8.2 Future Work

The proposed Traffic grouping approach based on individual attacks was implemented easily because of the CIDDS-001 and CIDDS-002 datasets. These datasets have set the standard for the IDS dataset and the detail that is to be provided In the dataset and documentation. The knowledge of individual attacks in the dataset help design better intrusion detection modules. The CIDDS-001 had very few attack records with the approach and features compared to other IDS datasets. As new datasets are released, this proposed method can be used to evaluate and the quality of the records in that dataset and improve the performance of the ML models.

The approach was not tested in a real-world environment and instead was tested on a similar CIDDS-002 dataset. The applicability of the approach in a real-world environment where NetFlow traffic is collected to a predetermined limit and then preprocessed and fed to the model to detect patterns.

The next goal of this approach is to detect the learned patterns from the incomplete and mixed record. This is a difficult task to be achieved using a single ML model. The use of multiple models that are trained and specialized in a single class can be used to replicate the behavior of humans. Self-learning behavior can be implemented by adding the misclassified records to the training dataset and re-training the model when a record is misclassified. During the retraining process, the model's hyperparameters can be tuned using Grid Search or Bayesian hyperparameters optimizer.

Appendix A

ML Model Performance

A.1 Shallow Learning models - Performance

Dataset	CIDDS-001						CIDDS-002					
	Binary			Multi			Binary			Multi		
Feature set	TPR	FPR	P	TPR	FPR	P	TPR	FPR	P	TPR	FPR	P
All	99.5	0.55	99.3	98.9	0.36	97.5	76.6	23.5	75	74.5	85.1	37.5
Subset -1	99.5	0.6	99.3	99.5	0.18	98.5	80.7	19.3	77.4	73.8	8.7	50
Subset -2	96.7	3.3	96.3	96.7	1.1	95.6	44.8	55.2	49.7	38.6	20.5	24.7
Subset - 3	97.8	2.2	97.4	95.6	4.6	91.2	48.3	51.7	65.5	33.8	22	27.9
Subset- 4	97.3	2.7	96.8	92.2	1.3	93.7	63.5	36.6	59.5	58.6	13.8	35.7
Subset - 5	98.4	1.6	98.4	99.5	1.3	95.7	75.1	24.8	67.4	63.5	12.2	37.6

Table A.1: Shallow Learning - KNN Performance

Dataset	CIDDS-001						CIDDS-002					
	Binary			Multi			Binary			Multi		
Feature set	TPR	FPR	P	TPR	FPR	P	TPR	FPR	P	TPR	FPR	P
All	99.5	0.6	99.3	99.5	0.2	98.8	99.3	0.7	98.6	96.6	1.15	66.7
Subset -1	99.5	0.6	99.3	99.5	0.2	99.1	99.3	0.7	98.6	95.2	1.6	49.8
Subset -2	99.5	0.6	99.3	98.9	0.4	98.3	97.2	2.8	94.7	91.1	2.9	59.5
Subset - 3	99.5	0.6	99.3	99.5	0.2	99.2	99.3	0.7	98.6	86.2	4.6	48.4
Subset- 4	99.5	0.6	99.3	99.5	0.2	99.8	95.9	4.1	92.5	77.2	7.6	50
Subset - 5	98.4	1.6	98.8	98.4	0.6	99.3	95.1	4.8	97	93.8	2.1	66.5

Table A.2: Shallow Learning - RF Performance

Dataset	CIDDS-001						CIDDS-002					
	Binary			Multi			Binary			Multi		
Feature set	TPR	FPR	P	TPR	FPR	P	TPR	FPR	P	TPR	FPR	P
All	99.5	0.55	99.3	99.5	0.18	99.5	99.3	0.69	98.6	96.6	1.2	66.7
Subset -1	99.5	0.55	99.3	99.5	0.18	99.5	99.3	0.69	98.6	93.1	2.3	66.7
Subset -2	98.9	0.2	98.9	98.9	0.36	98.9	73.1	26.9	63.8	75.1	8.1	44.4
Subset - 3	100	0	100	98.4	0.6	96.1	99.3	0.7	98.6	80.6	6.4	50
Subset- 4	98.4	1.6	98.4	99.2	0.18	99.8	71.1	28.9	62.5	83.5	5.5	56.1
Subset - 5	98.8	2.2	97.7	97.8	0.7	98.5	93.1	6.9	91.1	92.4	2.5	65.8

Table A.3: Shallow Learning - SVM Performance

Dataset	CIDDS-001						CIDDS-002					
	Binary			Multi			Binary			Multi		
Feature set	TPR	FPR	P	TPR	FPR	P	TPR	FPR	P	TPR	FPR	P
All	97.8	2.2	97.9	94.5	0.2	89.9	85.1	14.8	79.6	86.2	4.6	43.2
Subset -1	96.8	3.8	96.4	96.2	1.3	94.2	91	8.9	91.4	84.1	5.3	56.6
Subset -2	97.3	2.7	97.3	95.1	1.6	93.9	80	20	75.5	66.9	11.1	35.1
Subset - 3	93.9	6	93.7	94.5	0.18	92.7	88.9	11	83.9	71	9.7	42.9
Subset- 4	96.7	3.3	96.8	94.5	0.2	92.7	84.8	15.2	80	81.4	6.2	52.7
Subset - 5	93.4	6.6	93.8	95.6	1.6	94.2	92.4	7.6	93.8	80.7	6.4	39.2

Table A.4: Shallow Learning - DT Performance

A.2 Deep Learning Model - Performance

Dataset	CIDDS-001						CIDDS-002					
	Binary			Multi			Binary			Multi		
Feature set	TPR	FPR	P	TPR	FPR	P	TPR	FPR	P	TPR	FPR	P
All	98.9	1.1	98.9	97.8	0.7	99.1	99.3	6.9	98.6	97.2	0.9	66.7
Subset -1	98.9	1.1	99.1	98.9	0.4	98.9	90.3	9.6	88.6	90.3	3.2	61.9
Subset -2	99.5	0.6	99.3	98.3	0.6	96.8	62.8	37.2	68.3	61.4	12.8	41.9
Subset - 3	98.9	1.1	99.1	98.9	3.6	97.2	83.4	16.5	79.3	87.6	4.1	58.4
Subset- 4	100	0	100	99.4	0.2	99.8	83.4	16.5	77.9	66.9	11	43.1
Subset - 5	98.4	1.6	98.7	98.4	0.6	98.3	91.7	8.3	95.1	91	2.9	64.4

Table A.5: Deep Learning - CNN Performance

Dataset	CIDDS-001						CIDDS-002					
	Binary			Multi			Binary			Multi		
Feature set	TPR	FPR	P	TPR	FPR	P	TPR	FPR	P	TPR	FPR	P
All	99.4	0.6	99.3	98.9	0.4	98.3	95.2	4.8	94.6	95.2	1.6	64.5
Subset -1	99.5	0.6	99.3	98.9	0.4	98.9	97.9	2.1	97.6	89.7	3.5	49.3
Subset -2	100	0	100	99.5	0.2	99.1	85.5	14.8	82.6	83.5	5.5	57.4
Subset - 3	100	0	100	99.5	0.2	99.8	95.6	4.1	96.2	91.1	2.9	64.1
Subset- 4	100	0	100	98.9	0.4	99.5	88.3	11.7	91.0	78.6	7.1	47.1
Subset - 5	97.8	2.2	97.9	98.4	0.6	98.7	93.1	6.9	95.9	90.4	3.2	63.9

Table A.6: Deep Learning - DNN Performance

Dataset	CIDDS-001						CIDDS-002					
	Binary			Multi			Binary			Multi		
Feature set	TPR	FPR	P	TPR	FPR	P	TPR	FPR	P	TPR	FPR	P
All	99.5	0.6	99.6	99.5	0.2	98.3	85.5	14.8	79.6	92.4	2.5	64.5
Subset -1	98.9	1.1	99.1	98.9	0.2	98.1	93.1	6.9	95.7	82.8	5.5	61.6
Subset -2	100	0	100	99.5	0.2	99.2	84.8	15.1	88.4	57.2	14.3	38.4
Subset - 3	100	0	100	100	0	100	98.6	1.4	98.1	94.5	1.8	65.5
Subset- 4	100	0	100	98.4	0.6	98.8	64.9	35.2	67.3	71.7	9.4	45.5
Subset - 5	97.8	2.2	98.2	98.9	0.4	98.3	92.4	7.6	95.5	91.7	2.8	64.7

Table A.7: Deep Learning - RNN Performance

Dataset	CIDDS-001						CIDDS-002					
	Binary			Multi			Binary			Multi		
Feature set	TPR	FPR	P	TPR	FPR	P	TPR	FPR	P	TPR	FPR	P
All	100	0	100	99.5	0.2	99.1	100	0	100	93.1	6.8	65.2
Subset -1	100	0	100	99.5	0.2	99.1	93.1	6.9	95.9	93.1	6.8	65.2
Subset -2	99.5	0.6	99.3	98.9	0.4	98.3	60.7	39.3	60.1	83.4	16.5	55.1
Subset - 3	100	0	100	98.9	0.4	97.2	96.6	3.5	96.6	93.1	2.3	66.7
Subset- 4	100	0	100	98.9	0.4	99.6	88.9	11	93.7	91.7	2.8	64.7
Subset - 5	96.7	3.3	97.1	98.4	0.6	97.4	91.7	8.2	91.9	92.4	2.5	48

Table A.8: Deep Learning - LSTM Performance

Dataset	CIDDS-001						CIDDS-002					
	Binary			Multi			Binary			Multi		
Feature set	TPR	FPR	P	TPR	FPR	P	TPR	FPR	P	TPR	FPR	P
All	100	0	100	99.5	0.2	99.1	71.2	28.3	72.6	89.7	3.5	57.7
Subset -1	99.5	0.6	99.3	98.9	0.2	99.1	95.2	4.8	95.7	93.1	2.3	49.1
Subset -2	100	0	100	99.5	0.2	98.3	87.6	12.4	88.6	81.3	6.2	57
Subset - 3	100	0	100	100	0	100	95.7	4.1	96.1	86.2	4.6	63.1
Subset- 4	98.9	1.1	98.7	98.9	0.4	98.3	58.6	41.3	59.9	84.8	5.1	56.9
Subset - 5	98.4	1.6	98.4	99.5	0.2	99.2	91	9.1	92.9	91	2.9	64.2

Table A.9: Deep Learning - MLP Performance

Dataset	CIDDS-001						CIDDS-002					
	Binary			Multi			Binary			Multi		
Feature set	TPR	FPR	P	TPR	FPR	P	TPR	FPR	P	TPR	FPR	P
All	100	0	100	100	0	100	99.3	0.7	98.6	94.4	2.5	50
Subset -1	100	0	100	99.5	0.2	99.8	99.3	6.9	98.5	89.7	3.5	63.9
Subset -2	86.8	13.1	90.3	90.1	3.3	64.7	79.3	20.7	81.2	79.3	6.9	59.7
Subset - 3	100	0	100	90.1	3.2	63.2	99.3	0.7	98.6	89.7	3.5	65.1
Subset- 4	100	0	100	98.4	0.6	97.4	99.5	0.6	99.3	73.1	8.9	32.2
Subset - 5	98.3	1.6	98.4	98.9	0.4	98.9	94.5	5.5	94.1	91.8	2.9	48.7

Table A.10: Deep Learning - DBN Performance

A.3 Deep-Shallow Learning Model - Performance

Dataset	CIDDS-001						CIDDS-002					
	Binary			Multi			Binary			Multi		
Feature set	TPR	FPR	P	TPR	FPR	P	TPR	FPR	P	TPR	FPR	P
All	99.5	0.6	99.3	98.9	0.4	97.5	91	8.9	86.1	76.5	7.8	37.7
Subset -1	100	0	100	99.5	0.2	98.5	98.6	1.4	97.2	91	2.9	66.7
Subset -2	95.6	4.3	95.1	99.8	0.7	97.1	41.3	58.6	64.2	44.8	81.6	37.6
Subset - 3	100	0	100	100	0	100	95.9	4.4	96.2	82	5.9	47.5
Subset- 4	99.5	0.6	99.3	99.5	0.6	98.5	97.1	2.8	97.1	66.9	11	42.5
Subset - 5	69.4	30.1	76.7	98.4	0.6	97.2	86.2	13.8	80.4	63.5	12.2	30.6

Table A.11: Deep Learning - CNN-KNN Performance

Dataset	CIDDS-001						CIDDS-002					
	Binary			Multi			Binary			Multi		
Feature set	TPR	FPR	P	TPR	FPR	P	TPR	FPR	P	TPR	FPR	P
All	99.4	0.6	99.3	99.5	0.2	99.8	96.5	3.5	96.6	96.5	1.2	66.7
Subset -1	100	0	100	99.5	0.4	99.8	100	0	100	96.6	1.2	66.7
Subset -2	99.5	0.6	95.1	99.5	0.2	99.8	62.8	37.2	62.8	75.9	8.1	45.9
Subset - 3	99.5	5.5	99.3	98.7	0.6	98.1	97.1	2.8	97.1	89.7	3.5	66.7
Subset- 4	99.5	0.6	99.3	99.5	0.2	99.8	85.5	14.5	82.6	80	6.7	42.5
Subset - 5	99.5	0.6	99.3	99.5	0.2	99.8	93.7	6.1	94.6	91	2.9	64.5

Table A.12: Deep Learning - CNN-SVM Performance

Dataset	CIDDS-001						CIDDS-002					
	Binary			Multi			Binary			Multi		
Feature set	TPR	FPR	P	TPR	FPR	P	TPR	FPR	P	TPR	FPR	P
All	99.5	0.6	99.3	98.9	0.4	98.2	99.3	0.7	98.5	95.9	1.4	50
Subset -1	99.5	0.6	99.5	98.9	3.6	98.2	88.9	11	91.5	93.1	2.3	49.1
Subset -2	98.9	1.1	98.9	99.5	0.2	99.2	84.1	15.9	87.8	80.7	6.4	56.7
Subset - 3	99.5	5.5	99.3	100	0	100	95.7	4.8	95.7	91	2.9	63.5
Subset- 4	99.5	0.6	99.3	99.5	0.2	99.1	84.1	15.9	87.8	83.5	5.5	61.8
Subset - 5	97.8	2.2	97.9	96.2	1.3	97.1	93.1	6.9	94.2	88.3	3.9	49.7

Table A.13: Deep Learning - CNN-RF Performance

Dataset	CIDDS-001						CIDDS-002					
	Binary			Multi			Binary			Multi		
Feature set	TPR	FPR	P	TPR	FPR	P	TPR	FPR	P	TPR	FPR	P
All	98.9	1.1	98.9	99.5	0.2	98.5	82.1	17.9	75.1	80.7	6.4	52.9
Subset -1	98.9	1.1	98.7	98.9	3.6	98.2	88.9	11	86.3	89.7	3.5	60.8
Subset -2	99.5	0.6	99.3	98.9	0.4	98.5	86.9	13.1	86.4	78.5	7.1	49.4
Subset - 3	98.4	1.6	98	95.1	1.6	92.5	82.4	17.2	76.7	78.6	7.1	43.9
Subset- 4	98.9	1.1	98.7	98.9	0.4	98.8	84.4	15.2	82.8	86.2	4.6	43
Subset - 5	95.1	4.9	95.8	94.5	1.8	94.2	87.5	12.4	82.2	87.6	42.9	4.1

Table A.14: Deep Learning - CNN-DT Performance

Dataset	CIDDS-001						CIDDS-002					
	Binary			Multi			Binary			Multi		
Feature set	TPR	FPR	P	TPR	FPR	P	TPR	FPR	P	TPR	FPR	P
All	98.4	1.6	98	93.4	2.2	90.6	66.2	33.8	70.4	75.9	69.1	15.3
Subset -1	99.4	0.6	99.3	83.1	5.6	83.9	88.3	11.8	83.3	42.1	19.3	50
Subset -2	95.6	4.4	95.1	98.9	14.6	93.3	23.5	76.6	11.7	42.1	19.3	22.3
Subset - 3	97.8	2.2	97.4	98.4	5.5	96.9	71.7	28.3	72.7	83.4	5.5	58.1
Subset- 4	68.9	31.1	78.1	98.9	0.4	97.5	23.4	76.6	11.7	82.8	5.7	44.9
Subset - 5	89.6	1.0	88.9	90.1	3.3	88.3	87.6	12.4	82.3	84.1	3.3	47.3

Table A.15: Deep Learning - RNN-KNN Performance

Dataset	CIDDS-001						CIDDS-002					
	Binary			Multi			Binary			Multi		
Feature set	TPR	FPR	P	TPR	FPR	P	TPR	FPR	P	TPR	FPR	P
All	100	0	100	99.5	0.2	99.1	98.6	1.4	99.1	88.9	3.7	43.9
Subset -1	99.4	0.6	99.3	99.5	0.2	99.8	95.2	4.8	95.7	86.2	4.6	66.1
Subset -2	99.5	0.6	99.3	98.9	0.4	98.5	58.6	41.4	59.1	83.4	5.5	58.1
Subset - 3	100	0	100	98.9	0.4	97.5	97.9	2.1	95.4	89.7	3.5	63.6
Subset- 4	99.5	0.6	99.3	98.9	0.4	98.1	66.9	33.1	70.1	83.4	5.5	64.7
Subset - 5	98.9	1.1	98.7	98.9	0.4	98.3	93.8	6.2	94.6	91.7	2.8	65.2

Table A.16: Deep Learning - RNN-SVM Performance

Dataset	CIDDS-001						CIDDS-002					
	Binary			Multi			Binary			Multi		
Feature set	TPR	FPR	P	TPR	FPR	P	TPR	FPR	P	TPR	FPR	P
All	99.5	0.6	99.3	98.4	0.6	97.9	96.5	3.5	96.6	91	2.9	48.5
Subset -1	99.4	0.6	99.3	98.9	0.4	98.2	90.3	9.7	91.4	88.3	3.9	49.1
Subset -2	99.5	0.6	99.3	99.5	0.2	99.2	84.1	15.4	87.8	80.6	6.4	56.9
Subset - 3	100	0	100	99.5	0.2	99.1	99.3	0.7	98.1	91	2.9	63.5
Subset- 4	100	0	100	99.5	0.2	99.1	86.9	13.1	87.9	86.9	4.3	61.8
Subset - 5	97.8	2.2	97.9	97.3	0.9	98.2	91.7	8.3	93.4	90.3	3.2	64.2

Table A.17: Deep Learning - RNN-RF Performance

Dataset	CIDDS-001						CIDDS-002					
	Binary			Multi			Binary			Multi		
Feature set	TPR	FPR	P	TPR	FPR	P	TPR	FPR	P	TPR	FPR	P
All	94.5	0.6	94	94.5	0.2	89.7	87.6	12.4	82.1	56.6	14.5	33.5
Subset -1	94.5	5.5	93.9	97.8	0.7	96.6	80	20	72.1	75.2	8.3	38.2
Subset -2	98.4	1.6	98.4	94.5	6.1	92.1	79.3	20.7	76.9	73.1	8.9	32.4
Subset - 3	96.7	3.3	99.3	97.3	0.9	96.3	87.6	12.4	82.3	86.9	4.3	44.1
Subset- 4	96.1	3.8	96.4	95.1	1.6	93.9	79.3	20.7	70.9	76.6	7.8	34.1
Subset - 5	96.7	3.3	96.8	90.1	3.3	94	86.2	1.4	80.6	84.1	5.3	47.1

Table A.18: Deep Learning - RNN-DT Performance

Dataset	CIDDS-001						CIDDS-002					
	Binary			Multi			Binary			Multi		
Feature set	TPR	FPR	P	TPR	FPR	P	TPR	FPR	P	TPR	FPR	P
All	100	0	100	98.9	0.4	97.5	65.5	34.5	70.2	75.1	8.3	50
Subset -1	100	0	100	98.9	0.4	97.5	71.0	28.9	72.4	59.3	13.5	49.3
Subset -2	99.5	0.6	99.3	98.9	0.4	98.5	88.3	11.7	86.6	84.8	5.1	58.9
Subset - 3	100	0	100	99.5	0.2	98.5	88.9	11	83.8	48.2	17.9	63.3
Subset- 4	99.5	0.6	99.3	98.9	0.4	97.5	88.3	11.7	86.6	82.8	5.5	44.7
Subset - 5	98.3	1.6	98.7	97.8	0.7	97.1	91.7	8.3	93.4	79.3	6.9	49.9

Table A.19: Deep Learning - LSTM-KNN Performance

Dataset	CIDDS-001						CIDDS-002					
	Binary			Multi			Binary			Multi		
Feature set	TPR	FPR	P	TPR	FPR	P	TPR	FPR	P	TPR	FPR	P
All	99.5	0.6	99.3	99.5	0.2	99.8	84.1	15.9	78.3	96.6	1.1	66.7
Subset -1	100	0	100	99.5	0.2	99.8	99.3	0.7	99.6	93.1	2.3	65.8
Subset -2	99.5	0.6	99.3	98.9	0.4	99.6	83.5	16.6	77.9	76.5	7.8	46.7
Subset - 3	99.4	0.6	99.3	98.9	0.4	97.6	95.2	4.8	95.7	76.5	4.1	63.4
Subset- 4	100	0	100	98.9	0.4	99.5	92.4	7.6	92.5	80.7	6.4	51.3
Subset - 5	98.9	1.1	98.9	99.5	0.2	99.1	93.1	6.9	94.5	95.8	1.4	66.4

Table A.20: Deep Learning - LSTM-SVM Performance

Dataset	CIDDS-001						CIDDS-002					
	Binary			Multi			Binary			Multi		
Feature set	TPR	FPR	P	TPR	FPR	P	TPR	FPR	P	TPR	FPR	P
All	100	0	100	99.5	0.2	99.1	99.3	0.7	98.6	96.1	1.1	66.7
Subset -1	100	0	100	99.5	0.2	99.1	100	0	100	87.5	6.7	63.4
Subset -2	99.5	0.6	99.3	98.9	0.4	98.3	85.5	14.5	88.9	84.1	5.1	53.5
Subset - 3	99.5	0.6	99.3	98.9	0.4	98.2	94.5	5.5	95.2	84.8	5.1	63.4
Subset- 4	99.5	0.6	99.3	98.9	0.4	98.2	90.3	9.6	99.3	86.9	4.4	47.4
Subset - 5	98.9	1.1	98.9	98.9	0.4	98.9	97.2	2.8	98.3	93.8	2.1	65.6

Table A.21: Deep Learning - LSTM-RF Performance

Dataset	CIDDS-001						CIDDS-002					
	Binary			Multi			Binary			Multi		
Feature set	TPR	FPR	P	TPR	FPR	P	TPR	FPR	P	TPR	FPR	P
All	99.5	0.6	99.3	98.9	3.6	97.7	75	4.8	69.7	80	6.7	42.1
Subset -1	99.5	0.6	99.3	98.9	3.6	99.6	97.2	2.8	95.4	81.4	6.2	51.3
Subset -2	99.5	0.6	99.3	94.5	1.8	92.6	82.8	17.2	78.4	77.2	7.6	36.1
Subset - 3	99.5	0.6	99.3	98.9	0.4	98.2	90.3	9.6	87.8	79.3	6.9	39.7
Subset- 4	94.5	5.5	95.4	98.9	0.4	98.3	84.4	15.2	80.9	83.4	5.5	53.1
Subset - 5	96.2	3.8	96.1	97.8	2.1	97.9	95.9	4.1	94.2	88.3	3.9	56.6

Table A.22: Deep Learning - LSTM-DT Performance

A.4 Deep-Deep Learning Model - Performance

Dataset	CIDDS-001						CIDDS-002					
	Binary			Multi			Binary			Multi		
Feature set	TPR	FPR	P	TPR	FPR	P	TPR	FPR	P	TPR	FPR	P
All	100	0	100	99.5	0.2	99.8	99.3	0.7	98.6	96.6	1.2	66.7
Subset -1	99.4	0.6	99.6	99.5	0.2	99.1	94.5	5.5	96.6	96.5	1.2	66.7
Subset -2	100	0	100	98.4	0.6	96.1	84.1	15.9	87.8	85.3	5.5	59.7
Subset - 3	100	0	100	99.5	0.2	99.1	95.2	4.8	95.7	95.2	4.8	65.7
Subset- 4	99.5	0.6	99.3	99.5	0.2	99.1	88.3	11.7	91.1	71.3	9.6	46.4
Subset - 5	100	0	100	98.9	0.4	98.9	93.8	6.2	92.5	81.4	6.2	52.3

Table A.23: Deep Learning - CNN-RNN Performance

Dataset	CIDDS-001						CIDDS-002					
	Binary			Multi			Binary			Multi		
Feature set	TPR	FPR	P	TPR	FPR	P	TPR	FPR	P	TPR	FPR	P
All	99.4	0.6	99.3	95.6	1.5	91.1	99.3	0.7	98.5	67.6	10.8	47.1
Subset -1	99.4	0.6	99.3	98.9	0.4	98.2	95.2	4.8	95.7	96.5	1.2	66.7
Subset -2	99.4	0.6	99.3	98.9	0.4	98.3	75.2	24.7	69.2	82.8	5.8	58.7
Subset - 3	99.5	0.6	99.3	99.5	0.2	99.1	98.2	1.4	98.1	86.2	4.6	62.6
Subset- 4	99.5	0.6	99.3	99.5	0.2	99.7	88.9	11.3	89.7	80.7	6.4	50.9
Subset - 5	96.2	3.8	96.5	97.8	0.7	97.2	95.2	4.8	95.7	91	2.9	50

Table A.24: Deep Learning - CNN-LSTM Performance

Dataset	CIDDS-001						CIDDS-002					
	Binary			Multi			Binary			Multi		
Feature set	TPR	FPR	P	TPR	FPR	P	TPR	FPR	P	TPR	FPR	P
All	98.9	1.1	98.9	99.5	0.2	99.1	99.3	0.6	99.8	91	2.9	59.5
Subset -1	99.4	0.6	99.3	99.5	0.2	99.1	95.2	4.8	95.7	96.5	1.2	66.7
Subset -2	99.5	0.6	99.3	99.5	0.2	99.1	56.9	42.1	99.3	57.9	14	39.7
Subset - 3	99.5	0.6	99.3	99.5	0.2	98.5	94.5	5.5	95.2	91	2.9	63
Subset- 4	99.5	0.6	99.3	99.5	0.2	99.1	90.3	9.7	90.8	84.1	15.3	59.6
Subset - 5	100	0	100	98.4	0.6	97.4	95.2	4.8	97	91	2.9	48.5

Table A.25: Deep Learning - CNN-MLP Performance

Dataset	CIDDS-001						CIDDS-002					
	Binary			Multi			Binary			Multi		
Feature set	TPR	FPR	P	TPR	FPR	P	TPR	FPR	P	TPR	FPR	P
All	100	0	100	99.5	0.2	99.1	93.8	0.6	98.6	92.4	2.5	64.9
Subset -1	100	0	100	99.5	0.2	99.8	88.3	11.7	93.4	91	2.9	64.4
Subset -2	100	0	100	98.9	0.4	98.5	57.9	42.1	58.9	81.4	2.1	57.6
Subset - 3	99.5	5.5	99.3	99.5	0.2	99.1	99.3	6.9	98.6	94.5	5.5	65.2
Subset- 4	100	0	100	98.9	1.1	98.8	88.9	11	83.8	84.1	6.3	62.1
Subset - 5	98.4	1.6	98.4	98.9	0.6	98	95.2	4.8	97	93.1	2.3	65.2

Table A.26: Deep Learning - RNN-RNN Performance

Dataset	CIDDS-001						CIDDS-002					
	Binary			Multi			Binary			Multi		
Feature set	TPR	FPR	P	TPR	FPR	P	TPR	FPR	P	TPR	FPR	P
All	99.5	0.6	99.3	98.9	0.4	98.2	99.3	0.6	98.6	95.9	1.4	50
Subset -1	99.4	0.6	99.3	98.9	0.4	98.2	97.9	2.1	97.6	93.8	2.1	49.3
Subset -2	99.5	0.6	99.3	98.9	0.4	98.5	83.4	16.6	79.1	82.1	5.9	58.2
Subset - 3	99.5	5.5	99.3	98.4	0.6	96.3	97.6	2.1	97.6	89.7	3.5	65.5
Subset- 4	99.5	0.6	99.3	98.9	0.4	98.8	91.7	8.3	93.4	60.9	13.1	56.9
Subset - 5	96.2	3.8	96.4	96.7	1.1	97.9	84.8	15.1	79.1	90.3	3.2	48.1

Table A.27: Deep Learning - LSTM-LSTM Performance

Appendix B

Project Plan

B.1 Tasks, Milestones, and Timelines

Task	Begin Date	End Date	Duration (in days)
Begin F21RP	11/1/2021	8/4/2021	64
First Discussion with supervisor	11/1/2021	11/1/2021	1
Background review for Intrusion	19/1/2021	27/1/2021	7
Collecting research papers relevant to the project	21/1/2021	19/2/2021	22
Meeting with supervisor	22/1/2021	22/1/2021	1
Background review for Intrusion detection	2/2/2021	8/2/2021	5
Meeting with supervisor	16/2/2021	16/2/2021	1
Writing literature review of research papers	23/2/2021	1/3/2021	5
Selection of IDS dataset	19/2/2021	26/2/2021	6
Meeting with supervisor	1/3/2021	1/3/2021	1
Literature review of the Feature selection algorithm	2/3/2021	19/3/2021	14
Background research for IDS	4/3/2021	23/3/2021	14
Meeting with supervisor	8/3/2021	8/3/2021	1
Literature review of preprocessing datasets	11/3/2021	16/3/2021	4
Writing draft for literature review	12/3/2021	6/4/2021	18
Meeting with supervisor	15/3/2021	15/3/2021	1
Literature review of DL techniques	18/3/2021	31/3/2021	10
Meeting with supervisor	22/3/2021	22/3/2021	1
Meeting with supervisor	29/3/2021	29/3/2021	1
Writing other sections of RP report	31/3/2021	6/4/2021	5
Meeting with supervisor	5/4/2021	5/4/2021	1
Meeting with supervisor	7/4/2021	7/4/2021	1
Finished RP report	7/4/2021	7/4/2021	0

Table B.1: F21RP PROJECT PLAN

Task	Begin Date	End Date	Duration (in days)
Begin F21MP	15/4/2021	12/8/2021	86
F21MP- First meeting with supervisor	15/4/2021	15/4/2021	0
Familiarize with the datasets	19/4/2021	23/4/2021	5
Review Meeting with supervisor	22/4/2021	22/4/2021	1
Review Specifications	26/4/2021	26/4/2021	1
Preprocess the dataset	28/4/2021	28/4/2021	1
Evaluate feature selection algorithm - CFS	30/4/2021	4/5/2021	3
Evaluate feature selection algorithm - IG	6/4/2021	10/4/2021	3
Evaluate Feature Selection - AR	11/5/2021	13/5/2021	3
Evaluate Feature Selection - mutual information	14/5/2021	8/5/2021	3
Select best feature selection algorithm	20/5/2021	20/5/2021	1
Dissertation draft writing	20/5/2021	10/8/2021	59
Evaluation of DL and ML algorithms	24/5/2021	6/8/2021	55
Evaluate Shallow ML algorithm - KNN	24/5/2021	26/5/2021	2
Evaluate DL model - DNN	24/5/2021	28/5/2021	5
Evaluate Shallow ML algorithm - DT	28/5/2021	1/6/2021	3
Evaluate DL algorithm - CNN	31/5/2021	4/6/2021	5
Evaluate shallow learning method - SVM	9/6/2021	11/6/2021	3
Evaluate DL algorithm - RNN	9/6/2021	15/6/2021	5
Evaluate DL algorithm - DBN	16/6/2021	22/6/2021	5
Evaluate DL algorithm - Autoencoder	23/6/2021	30/6/2021	6
Evaluate and compare different models' performance	2/7/2021	2/7/2021	1
Evaluation of Combinations of DL and ML algorithms	5/7/2021	5/8/2021	24
(Optional)Evaluation of combinations of DL techniques	5/7/2021	5/8/2021	24
Design GUI module	13/7/2021	30/7/2021	14
Design Evaluator module	28/7/2021	6/8/2021	8
Combine Evaluator and GUI modules	10/8/2021	12/8/2021	2
Review project and Dissertation	10/8/2021	12/8/2021	3
Submit Dissertation	13/8/2021	13/8/2021	1

Table B.2: F21MP PROJECT PLAN

The Gantt Charts were made using 'Gantt Project'.

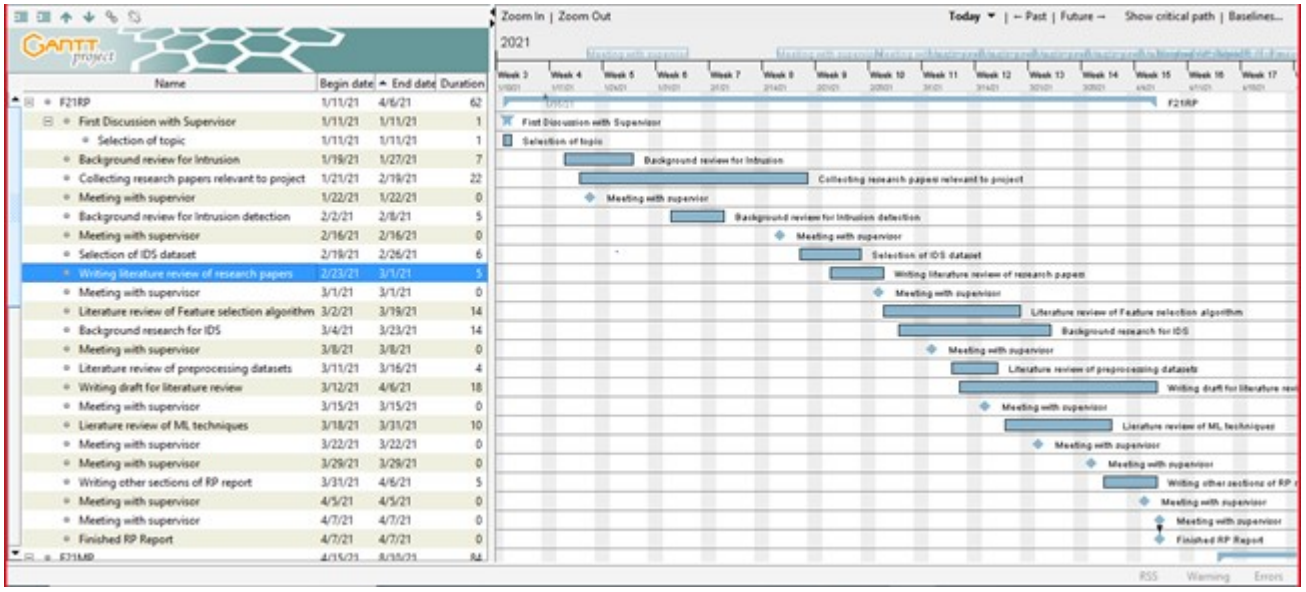


FIGURE B.1: F21RP GANTT CHART

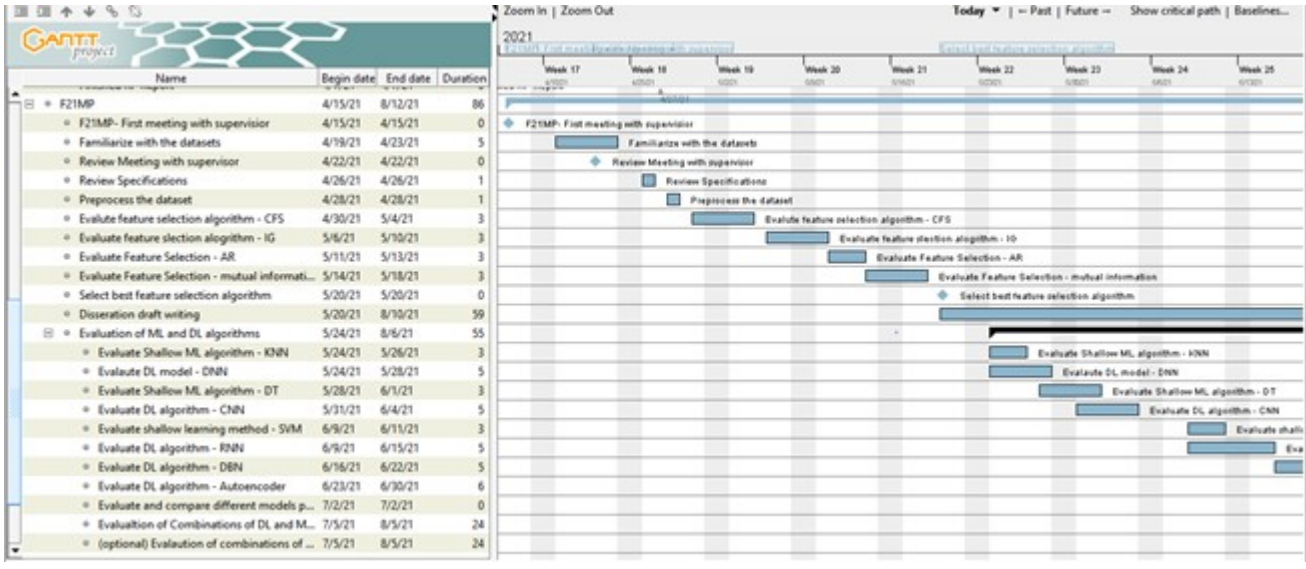


FIGURE B.2: F21MP GANTT CHART (1)

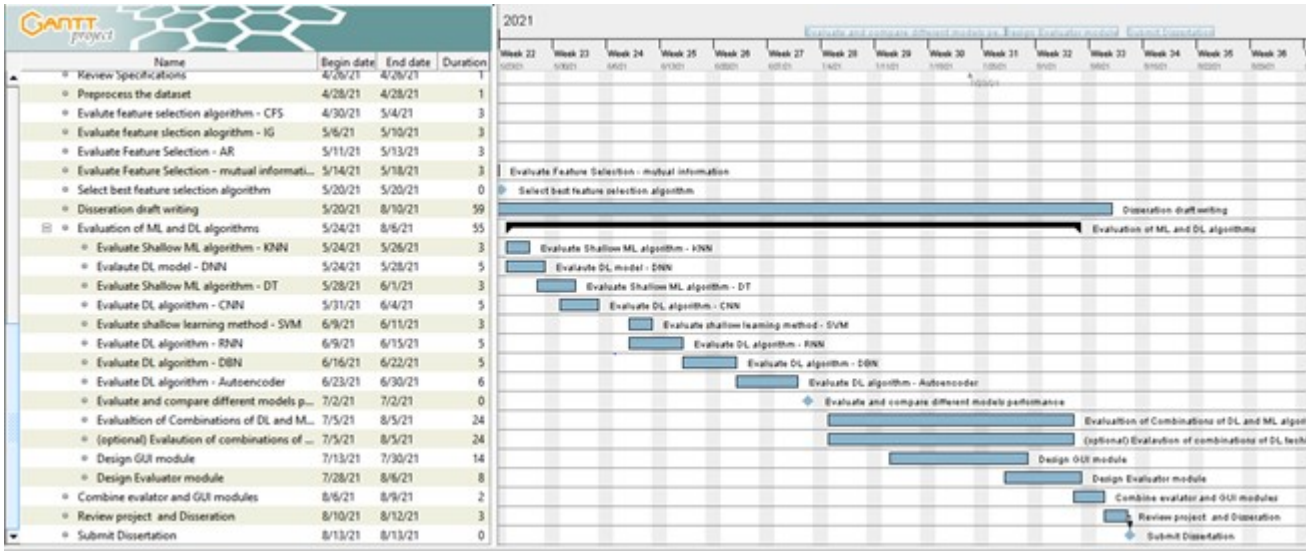


Figure B.3: F21MP GANTT CHART (2)

B.2 Risk Analysis

Risk	Likelihood	Impact	Result	Mitigation
The student unable to attend meetings (physically/digitally)	Medium	High	Delay	Inform supervisor about reason, preferably before the meeting. Reschedule the meeting if possible or ask for more time during the next allotted meeting
Supervisor unable to attend meetings (physically/digitally)	Medium	High	Delay	Reschedule meeting to if possible or ask for more time during next allotted meeting
Computational Restriction for project	Medium	Medium	Pauses progress	Extra computational resources can be obtained using Google Collab
Health Issues - Author	Medium	Medium	Depending on the severity, can cause delays	Leading of a healthy lifestyle, if health issues arise, change project plan according, ask for an extension if severe.

Falling behind schedule	High	High	Delay	Refactor project plan by allotting time for delayed tasks after analysis of priority of tasks.
Loss of source code or report	Low	High	Start over	Maintain a backup of both source code and report. For source code, GitHub can be used, and for report Google drive
Dataset Issue	Low	Medium	Pauses progress, Delay	Repeat feature selection process and model architecture process. Dataset can also be changed
Change to requirements of projects	Low	High	Changes project plan	Consult supervisor and refactor project plan to accommodate changes.

Table B.3: RISK ASSESSMENT

Bibliography

(2021). Information gain computation.

Abbott, R. P., Chin, J. S., Donnelley, J. E., Konigsford, W. L., Tokubo, S., and Webb, D. A. (1976). Security analysis and enhancements of computer operating systems. Technical report, NATIONAL BUREAU OF STANDARDS WASHINGTONDC INST FOR COMPUTER SCIENCES AND ....

Abt, S. and Baier, H. (2016). Are We Missing Labels? A Study of the Availability of Ground-Truth in Network Security Research. In *Proceedings - 3rd International Workshop on Building Analysis Datasets and Gathering Experience Returns for Security, BADGERS 2014*, pages 40–55. Institute of Electrical and Electronics Engineers Inc.

albertbup (2017). A python implementation of deep belief networks built upon numpy and tensorflow with scikit-learn compatibility.

Alrawashdeh, K. and Purdy, C. (2017). Toward an Online Anomaly Intrusion Detection System Based on Deep Learning. pages 195–200. Institute of Electrical and Electronics Engineers (IEEE).

Amiri, F., Rezaei Yousefi, M., Lucas, C., Shakery, A., and Yazdani, N. (2011). Mutual information-based feature selection for intrusion detection systems. *Journal of Network and Computer Applications*, 34(4):1184–1199.

Anderson, J. P. (1980). Computer security threat monitoring and surveillance. *ci.nii.ac.jp*.

Azad, T. (2008). *Securing Citrix XenApp Server in the Enterprise*.

Bace, R. (2000). Intrusion Detection. Technical report.

Bace, R. and Mell, P. (2001). NIST Special Publication on Intrusion Detection Systems. Technical report.

Berman, D., Buczak, A., Chavis, J., and Corbett, C. (2019). A Survey of Deep Learning Methods for Cyber Security. *Information*, 10(4):122.

Bisbey, Richard and Hollingworth, D. (1978). Protection analysis: Final report. *ISI/SR-78-13, Information Sciences Inst*, 3.

Buczak, A. L. and Guven, E. (2016). A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection. *IEEE Communications Surveys and Tutorials*, 18(2):1153–1176.

Chae, H.-S., Jo, B.-O., Choi, S.-H., and Park, T.-K. (2013). Feature Selection for Intrusion Detection using NSL-KDD. *Recent advances in computer science*, 20132:184–187.

Cohen, S. (2020). *Artificial Intelligence and Deep Learning in Pathology E-Book*. Elsevier Health Sciences.

Cordero, C. G., Vasilomanolakis, E., Milanov, N., Koch, C., Hausheer, D., and Muhlhauser, M. (2015). ID2T: A DIY dataset creation toolkit for Intrusion Detection Systems. In *2015 IEEE Conference on Communications and NetworkSecurity, CNS 2015*, pages 739–740. Institute of Electrical and Electronics Engineers Inc.

Cordero, C. G., Vasilomanolakis, E., Wainakh, A., Mühlhäuser, M., and Nadjm-Tehrani, S. (2021). On Generating Network Traffic Datasets with Synthetic Attacks for Intrusion Detection. *ACM Transactions on Privacy and Security*, 24(2):1–39.

Corsini, A., Yang, S. J., and Apruzzese, G. (2021). On the evaluation of sequential machine learning for network intrusion detection. *arXiv preprint arXiv:2106.07961*.

Creech, G. and Hu, J. (2013). Generation of a new IDS test dataset: Time to retire the KDD collection. In *IEEE Wireless Communications and Networking Conference, WCNC*, pages 4487–4492.

Cunningham, R. K., Lippmann, R. P., Fried, D. J., Garfinkel, S. L., Graf, I., Kendall, K. R., Webster, S. E., Wyschogrod, D., and Zissman, M. A. (1999). Evaluating Intrusion Detection Systems without Attacking your Friends: The 1998 DARPA Intrusion Detection Evaluation. Technical report, MASSACHUSETTS INST OF TECH LEXINGTON LINCOLN LAB.

Denning, D. and Neumann, P. (1985). *Requirements and model for IDES-a real-time intrusion-detection expert system*, volume 8. SRI International Menlo Park.

Doshi, R., Apthorpe, N., and Feamster, N. (2018). Machine learning ddos detection for consumer internet of things devices. In *2018 IEEE Security and Privacy Workshops (SPW)*, pages 29–35.

Dwarampudi, M. and Reddy, N. (2019). Effects of padding on lstms and cnns. *arXiv preprint arXiv:1903.07288*.

Farahani, G. (2020). Feature Selection Based on Cross-Correlation for the Intrusion Detection System. *Security and Communication Networks*, 2020.

Ferrag, M. A., Maglaras, L., Moschoyiannis, S., and Janicke, H. (2020). Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study. *Journal of Information Security and Applications*, 50:102419.

Gamage, S. and Samarabandu, J. (2020). Deep learning methods in network intrusion detection: A survey and an objective comparison. *Journal of Network and Computer Applications*, 169:102767.

Gao, N., Gao, L., Gao, Q., and Wang, H. (2015). An Intrusion Detection Model Based on Deep Belief Networks. In *Proceedings - 2014 2nd International Conference on Advanced Cloud and Big Data, CBD 2014*, pages 247–252. Institute of Electrical and Electronics Engineers Inc.

Gharib, A., Sharafaldin, I., Lashkari, A. H., and Ghorbani, A. A. (2017). An Evaluation Framework for Intrusion Detection Dataset. In *ICISS 2016 - 2016 International Conference on Information Science and Security*. Institute of Electrical and Electronics Engineers Inc.

Gil, C., Baños, R., Montoya, F. G., Gómez, J., Gil, C., Bañ, R., Márquez, . L., Montoya, F. G., and Montoya, . G. (2013). A Pareto-based Multi-objective Evolutionary Algorithm for Automatic Rule Generation in Network Intrusion Detection Systems. *Springer*, 17(2):255–263.

Gotterbarn, D., Miller, K., and Rogerson, S. (1997). Software engineering code of ethics. *Commun. ACM*, 40(11):110–118.

Gumusbas, D., Yldrm, T., Genovese, A., and Scotti, F. (2020). A Comprehensive Survey of Databases and Deep Learning Methods for Cybersecurity and Intrusion Detection Systems. *IEEE Systems Journal*, pages 1–15.

Hansman, S. and Hunt, R. (2005). A taxonomy of network and computer attacks. *Computers and Security*, 24(1):31–43.

Hindy, H., Brosset, D., Bayne, E., Seeam, A. K., Tachtatzis, C., Atkinson, R., and Bellekens, X. (2020). A Taxonomy of Network Threats and the Effect of Current Datasets on Intrusion Detection Systems. *IEEE Access*, 8:104650–104675.

Hodo, E., Bellekens, X., Hamilton, A., Tachtatzis, C., and Atkinson, R. (2017). Shallow and Deep Networks Intrusion Detection System: A Taxonomy and Survey. *arXiv*.

Ingre, B. and Yadav, A. (2015). Performance analysis of NSL-KDD dataset using ANN. In *International Conference on Signal Processing and Communication Engineering Systems - Proceedings of SPACES 2015, in Association with IEEE*, pages 92–96. Institute of Electrical and Electronics Engineers Inc.

Jabez, J. and Muthukumar, B. (2015). Intrusion detection system (ids): Anomaly detection using outlier detection approach. In *Procedia Computer Science*, volume 48, pages 338–346. Elsevier B.V.

Jones, C. (2010). *Software Engineering Best Practices: Lessons from Successful Projects in the Top Companies*. McGraw-Hill Education, New York.

Kanimozhi, V. and Prem Jacob, T. (2019). Artificial intelligence based network intrusion detection with hyper-parameter optimization tuning on the realistic cyber dataset CSE-CIC-IDS2018 using cloud computing. In *Proceedings of the 2019 IEEE International Conference on Communication and Signal Processing, ICCSP 2019*, pages 33–36. Institute of Electrical and Electronics Engineers Inc.

Kendall, K. (1999). A database of computer attacks for the evaluation of intrusion detection systems.

Khan, R. U., Zhang, X., Alazab, M., and Kumar, R. (2019). An improved convolutional neural network model for intrusion detection in networks. In *Proceedings - 2019 Cybersecurity and Cyberforensics Conference, CCC 2019*, pages 74–77. Institute of Electrical and Electronics Engineers Inc.

Kumar, D Ashok and Venugopalan, S. (2017). INTRUSION DETECTION SYSTEMS: A REVIEW. *International Journal of Advanced Research in Computer Science*, 8(8).

Kunhare, N., Tiwari, R., and Dhar, J. (2020). Particle swarm optimization and feature selection for intrusion detection system. *Sādhanā*, 45:1–14.

Lazarevic, A., Kumar, V., and Srivastava, J. (2005). Intrusion Detection: A Survey. In *Managing Cyber Threats*, pages 19–78. Springer-Verlag.

Lee, W. and Stolfo, S. J. (1998). Data Mining Approaches for Intrusion Detection. Technical report.

Leevy, J. L. and Khoshgoftaar, T. M. (2020). A survey and analysis of intrusion detection models based on CSE-CIC-IDS2018 Big Data. *Journal of Big Data*, 7(1):104.

Li, Y. and Zhang, B. (2019). An intrusion detection model based on multi-scale CNN. In *Proceedings of 2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference, ITNEC 2019*, pages 214–218. Institute of Electrical and Electronics Engineers Inc.

Li, Z., Qin, Z., Huang, K., Yang, X., and Ye, S. (2017). Intrusion detection using convolutional neural networks for representation learning. In *International conference on neural information processing*, pages 858–866. Springer.

Liao, H.-J., Lin, C.-H. R., Lin, Y.-C., and Tung, K.-Y. (2012). Intrusion detection system: A comprehensive review. *Journal of Network and Computer Applications*, 36:16–24.

Lopez-del Rio, A., Martin, M., Perera-Lluna, A., and Saidi, R. (2020). Effect of sequence padding on the performance of deep learning models in archaeal protein functional prediction. *Scientific reports*, 10(1):1–14.

Moxley-Wyles, B., Colling, R., and Verrill, C. (2020). Artificial intelligence in pathology: an overview.

Niyaz, Q., Sun, W., Javaid, A. Y., and Alam, M. (2016). A Deep Learning Approach for Network Intrusion Detection System. *Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies (formerly BIONETICS)*, pages 21–26.

Panigrahi, R. and Borah, S. (2018). A detailed analysis of CICIDS2017 dataset for designing Intrusion Detection Systems IEEE International Conference on Advanced Computational and Communication Paradigms (ICACCP-2017) View project Analysis of Selected Clustering Algorithms Used in Intrusion Detection Systems View project A detailed analysis of CICIDS2017 dataset for designing Intrusion Detection Systems. *International Journal of Engineering Technology*, 7:479–482.

Patel, K. K. and Buddhadev, B. V. (2012). An Architecture of Hybrid Intrusion Detection System. *International Journal of Information and Network Security (IJINS)*, 2(2):1–6.

Pawlicki, M., Choraś, M., and Kozik, R. (2020). Defending network intrusion detection systems against adversarial evasion attacks. *Future Generation Computer Systems*, 110:148–154.

Potluri, S., Ahmed, S., and Diedrich, C. (2018). Convolutional neural networks for multi-class intrusion detection system. In *International Conference on Mining Intelligence and Knowledge Exploration*, pages 225–238. Springer.

Ring, M., Wunderlich, S., Grüdl, D., Landes, D., and Hotho, A. (2017). Flow-based benchmark data sets for intrusion detection. In *Proceedings of the 16th European Conference on Cyber Warfare and Security. ACPI*, pages 361–369.

Roberto Di Pietro and Luigi V. Mancini (2008). *Intrusion Detection Systems*, volume 38 of *Advances in Information Security*. Springer US, Boston, MA, 1 edition.

Rosay, A., Carlier, F., and Leroux, P. (2020). MLP4NIDS: An Efficient MLP-Based Network Intrusion Detection for CICIDS2017 Dataset. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 12081 LNCS, pages 240–254. Springer.

Shao, E. (2019). *Encoding IP Address as a Feature for Network Intrusion Detection*. PhD thesis, Purdue University Graduate School West Lafayette, Indiana.

Sharafaldin, I., Gharib, A., Lashkari, A. H., and Ghorbani, A. A. (2017). Towards a Reliable Intrusion Detection Benchmark Dataset. *Software Networking*, 2017(1):177–200.

Sharafaldin, I., Lashkari, A. H., and Ghorbani, A. A. (2018a). Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization. *ICISSp*, pages 108–116.

Sharafaldin, I., Lashkari, A. H., and Ghorbani, A. A. (2018b). Toward generating a new intrusion detection dataset and intrusion traffic characterization. *ICISSp*, 1:108–116.

Shenfield, A., Day, D., and Ayesh, A. (2018). Intelligent intrusion detection systems using artificial neural networks. *ICT Express*, 4(2):95–99.

Shone, N., Ngoc, T. N., Phai, V. D., and Shi, Q. (2018). A Deep Learning Approach to Network Intrusion Detection. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2(1):41–50.

Singh Panwar, Shailesh and Raiwani, YP and Panwar, L. S. (2019). Evaluation of network intrusion detection with features selection and machine learning algorithms on CICIDS-2017 dataset. *International Conference on Advances in Engineering Science Management & Technology (ICAESMT)-2019, Uttaranchal University, Dehradun, India*.

Tang, T. A., Mhamdi, L., McLernon, D., Zaidi, S. A. R., and Ghogho, M. (2016). Deep learning approach for Network Intrusion Detection in Software Defined Networking. In *Proceedings - 2016 International Conference on Wireless Networks and Mobile Communications, WINCOM 2016: Green Communications and Networking*, pages 258–263. Institute of Electrical and Electronics Engineers Inc.

Tavallaee, M., Bagheri, E., Lu, W., and Ghorbani, A. A. (2009). A detailed analysis of the KDD CUP 99 data set. In *IEEE Symposium on Computational Intelligence for Security and Defense Applications, CISDA 2009*.

Viegas, E. K., Santin, A. O., and Oliveira, L. S. (2017). Toward a reliable anomaly-based intrusion detection in real-world environments. *Computer Networks*, 127:200–216.

Vinayakumar, R., Alazab, M., Soman, K. P., Poornachandran, P., Al-Nemrat, A., and Venkatraman, S. (2019). Deep Learning Approach for Intelligent Intrusion Detection System. *IEEE Access*, 7:41525–41550.

Vinayakumar, R., Soman, K., and Poornachandran, P. (2017). Applying convolutional neural network for network intrusion detection. In *2017 International Conference on*

Advances in Computing, Communications and Informatics (ICACCI), pages 1222–1228. IEEE.

Wu, S. X. and Banzhaf, W. (2010). The Use of Computational Intelligence in Intrusion Detection Systems: A Review. Technical report.

Xin, Y., Kong, L., Liu, Z., Chen, Y., Li, Y., Zhu, H., Gao, M., Hou, H., and Wang, C. (2018). Machine Learning and Deep Learning Methods for Cybersecurity. *IEEE Access*, 6:35365–35381.

Xu, C., Shen, J., Du, X., and Zhang, F. (2018). An Intrusion Detection System Using a Deep Neural Network with Gated Recurrent Units. *IEEE Access*, 6:48697–48707.

Yin, C., Zhu, Y., Fei, J., and He, X. (2017). A Deep Learning Approach for Intrusion Detection Using Recurrent Neural Networks. *IEEE Access*, 5:21954–21961.

Yong, L. and Bo, Z. (2019). An intrusion detection model based on multi-scale cnn. In *2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*, pages 214–218. IEEE.

ZixiaoShen (2019). Correlation-based-feature-selection. https://github.com/ZixiaoShen/Correlation-based-Feature-Selection.