Transferability of Intrusion Detection Systems Using Machine Learning between Networks

William Peter Mati
University of Windsor

Transferability of Intrusion Detection Systems Using Machine Learning Between Networks

By

William Peter Mati

A Thesis
Submitted to the Faculty of Graduate Studies
through the Department of Electrical and Computer Engineering
in Partial Fulfillment of the Requirements for
the Degree of Master of Applied Science
at the University of Windsor

Windsor, Ontario, Canada

2021

Transferability of Intrusion Detection Systems Using Machine Learning Between Networks

by

William Peter Mati

APPROVED BY:

____
L. Rueda
School of Computer Science

____
M. Mirhassani
Department of Electrical and Computer Engineering

____
K. Tepe, Advisor
Department of Electrical and Computer Engineering

October 4, 2021

DECLARATION OF ORIGINALITY

I hereby certify that I am the sole author of this thesis and that no part of this thesis has been published or submitted for publication.

I certify that, to the best of my knowledge, my thesis does not infringe upon anyone's copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my thesis, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that I have included copyrighted material that surpasses the bounds of fair dealing within the meaning of the Canada Copyright Act, I certify that I have obtained a written permission from the copyright owner(s) to include such material(s) in my thesis and have included copies of such copyright clearances to my appendix.

I declare that this is a true copy of my thesis, including any final revisions, as approved by my thesis committee and the Graduate Studies office, and that this thesis has not been submitted for a higher degree to any other University or Institution.

ABSTRACT

Intrusion detection systems (IDS) using machine learning is a next generation tool to strengthen the cyber security of networks. Such systems possess the potential to detect zero-day attacks, attacks that are unknown to researchers and are occurring for the first time in history. This thesis tackles novel ideas in this research domain and solves foreseeable issues of a practical deployment of such tool.

The main issue addressed in this thesis are situations where an entity intends to implement an IDS using machine learning onto their network, but do not have attack data available from their own network to train the IDS. A solution is to train the IDS using attack data from other networks. However, there is a degree of uncertainty whether this is feasible as different networks use different applications and have different uses. Such IDS may not be able to adequately operate on a network when trained on data from an entirely different network. The proposed methodology in this research recommends the training set should combine attack data collected from other networks with benign traffic which originates from the network the IDS is to be implemented on. This method is compared with a training set which is completely composed of both attack and benign data from a completely different network. The best performing model implemented with both training sets demonstrated the feasibility of both scenarios. Both versions of that model achieved an F1 score of 0.82 and 0.81 respectively, and both versions detected roughly 70% of attacks and 99% of benign traffic. However, most IDSs trained on the former training set listed yielded the best results. The main benefit of training a model on target network benign data is to minimize false positive classifications. The average model witnessed a 113% boost in precision, compared to their counterparts trained on foreign network benign data. Another issue addressed in this thesis is the detection scope of attacks. The IDS scope of detection is limited to the attacks it is trained on. Using the proposed IDS training set, an intuitive feature selection scheme and classification threshold adjustment, this thesis improves the IDS scope of detection to detect attacks outside of its training data. Feature selection can manipulate an IDS to detect specific attacks not included in its training data. Using threshold tuning, the IDSs in this thesis detected up to 200% more attacks. Both issues and solutions are simulated and verified in two separate scenarios using neural networks and random forest.

ACKNOWLEDGEMENTS

I would like to thank my supervisor, Dr. Kemal Tepe for this opportunity, and the knowledge and guidance he shared with me during my time in the WiCIP lab. I also wish to express my gratitude to my committee members, Dr. Luis Rueda, and Dr. Mitra Mirhassani for their valued comments and suggestions.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS/SYMBOLS

Abbreviation	**Meaning**
IDS	Intrusion Detection System
ML	Machine Learning
SMOTE	Synthetic Minority Oversampling Technique
NIDS	Network IDS
IPS	Intrusion Prevention System
AP	Access Point
NMS	Network Management System
MITM	Man In The Middle
DoS	Denial of Service
DDoS	Distributed DoS
BCE	Binary Cross Entropy
AUC	Area Under Curve
ROC	Receiver Operating Characteristic
ReLU	Rectified Linear Unit
RLROP	Reduce LR On Plateau
ES	Early Stopping
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
TTL	Time To Live
SSH	Secure Shell
FTP	File Transfer Protocol
XSS	Cross (X) Site Scripting
HTTP	Hypertext Transfer Protocol
ICMP	Internet Control Message Protocol
RBOA	Replaced Benign Original Anomaly
OBOA	Original benign Original Anomaly
SVM	Support Vector Machine
RF	Random Forest

Chapter 1 — **Introduction**

1.1. **Motivation**

Since the inception of the personal computer, technology has progressively been more integrated into everyday life. Especially due to the COVID-19 pandemic, both organizations and individuals have become more digitally integrated than ever before. In this digital world, a significant portion of everyday business operations is reliant on some form of information technology infrastructure, which has introduced a series of new opportunities for society to prosper. Unfortunately, this mass digitization has also led to an increased volume of cyber-attacks. A successful cyber-attack on the correct organization can result in billions of dollars in damages for governments and societies, thus it is crucial to mitigate such attacks. Cyber threats are constantly evolving, which may be difficult to defend against when they are first executed. This is called a "zero-day attack". One potential tool which may provide organizations and researchers to ability to stop and disarm such an attack is an intrusion detection system (IDS).

An IDS is a frontline defense mechanism against cyber-attacks. As either a software application within a host, or a dedicated device within a computer network, it functions as an alert system to scan incoming network traffic and give immediate warning of incoming attacks. This tool gives network administrators the opportunity to quickly stop an attack from being successfully executed as soon as it occurs within their network. The current IDS standard is signature based, which one key limitation is the inability to detect zero-day attacks. This weakness can be addressed by using a machine learning based IDS, which offers a wider scope to detect cyber-attacks.

The application of machine learning algorithms into IDS is an ongoing research field, which offers immense potential to upgrade current IDS methods. Instead of searching for specific attacks in network traffic, a machine learning IDS analyzes general traffic behaviors and distinguishes suspicious traffic which may have malicious objectives. This allows IDSs to become more robust, stop zero-day attacks and enhance network automation.

1.2. **Problem Statement**

The current standard for machine learning IDS is network dependent, where the data used to train the IDS originates from the same network the IDS is intended to be implemented on. A key requirement to develop such IDS, is the collection of large volumes of benign (non-malicious) and anomalous (malicious/attack) traffic data from a single network. Benign traffic is abundant as most network traffic is benign and is thus easy to collect. Collection of anomalous traffic data on the other hand, may be logistically difficult to collect from a single, specific network in practical scenarios. There are ultimately two options available to collect anomalous traffic data. The first option is to simulate attacks on the IDS target network. For commercial networks that need to constantly be fully operational, this option is undesirable as the networks normal operation will be interrupted by the simulated attacks, and thus network downtime will need to be scheduled. The second option to collect anomalous traffic data is to wait for real attacks to occur within the desired network. This option is also undesirable in the manner that it counterintuitive and time consuming. Ideally, real network attacks should be mitigated. In the case that a single network can collect data from real attacks, the duration of time to collect a sufficient volume of anomalous data with a diverse variety of network attacks is unknown and may potentially take years. This is also true for simulated attacks: it may take too long to simulate large volumes of attacks of different varieties. Furthermore, both options pose the potential risk of causing damage to the network, a risk all network administrators and organizations will not chance. Ultimately, in realistic scenarios, it is a challenge to collect anomalous traffic data, and thus restricts the ability to develop a strong machine learning IDS. Instead, IDSs should use attack data from a global repository or from other networks. However, no work has ever been done to study the feasibility of this, and how an IDS trained on data from foreign networks will interpret the traffic of the target network.

The main doubt is transferability, because of the varying applications used on different networks, and the different type of networks. For instance, it is unknown how an IDS trained on data from an IoT network used for sensor telemetry would interpret benign network traffic on a hardwired corporate office network. Most of the benign traffic in the corporate office network might be composed of HTTP requests, whereas most of the benign traffic in the telemetry network might consist of other protocols from a proprietary application. Another concern is the interpretability of attack data. If the exact same attack occurred on both networks, and an IDS is trained on the attack data from the telemetry network, then the question is whether the IDS would be able to detect that attack if deployed on the corporate office network.

1.3. **Thesis Contribution**

This research evaluates whether network traffic from foreign networks can be treated universally for training an IDS using machine learning and proposes a scheme to optimize IDS detection for such scenarios. Practically, this can be very beneficial in situations where entities wish to deploy a machine learning IDS on a network, but do not have any collected attack data from that network. An alternate solution for those entities is to use data from other networks, or data from a global repository. Hence, this research aims to determine whether an IDS trained on data from a foreign network will be able to detect attacks on a specific network.

Two separate machine learning algorithms are used to demonstrate the effectiveness of this scheme, deep neural networks, and random forest. The datasets used to train the models are large and contain a diverse variety of different cyber-attacks. The IDS models trained can detect similar and new attack tools it hasn't seen before during that the training stage.

The proposed IDS lays out a procedure to generate a training set for an IDS using network data from foreign networks. The ideal training set uses benign data from the network the IDS will be implemented on, and attack data from a foreign network. This thesis shows this method yields optimal results for detecting attacks. It is shown an IDS trained exclusively on benign and attack data from foreign networks can interpret network traffic in a new network but is not completely reliable. Instead, the proposed scheme addresses and solves the main drawbacks of an IDS trained completely on foreign network traffic.

Finally, this thesis simulates a realistic scenario where a trained IDS is faced with cyber-attacks it has never seen before. A methodology is explored to assist the IDS to detect such attacks, consisting of training an IDS on the proposed training set, alongside tuning the classification threshold using the ROC curve, feature selection and cost sensitive learning.

1.4. **Thesis Structure**

There are five main sections in this thesis. Chapter 1 discusses the importance and motivation behind this thesis, in addition to the contributions provided. Chapter 2 reviews the preliminary technical background required to understand the work done in this thesis, as well as literature review and the current state of IDS research. Chapter 3 discusses the methodology and approach to perform the work in this thesis. Chapter 4 presents the simulated results for the tested IDSs. Chapter 5 includes the conclusion and future work.

Chapter 2 — **Background & Literature Review**

2.1. Intrusion Detection System

This section introduces IDSs and provides fundamental knowledge about IDS types and operation.

2.1.1. Basic Operation

IDSs are implemented in a network as either a software application running on a host in the network, or as a dedicated device connected in the network. The basic function of an IDS is to inspect traffic and to detect malicious activity. If the IDS detects a suspicious packet, then the IDS will alert the network administrator. The network administrator via a network management system (NMS) device will review the marked traffic logs, and manually decide whether to act. Such actions may include restricting or allowing traffic to/from the destination, altering firewall rules, etc. This tactic is called "Passive Alerting and Manual Response".

An extension of an IDS is IPS, which has an active response to a potential threat. In addition to generating an alarm for the network administrator, an IPS will react to the threat without human intervention, usually from a decision table [1]. IPSs have two methods to defend against threats: reactive and proactive responses. The difference between the two is reactive responses take immediate action upon detected threats, and proactive responses are actions done prior to deter and mitigate an attack. IPS development is a separate area of research in the network security and automation field and is out of scope for this thesis.

In terms of network architecture for dedicated IDS devices, it is located behind the network firewall at a network access point. Figure 1 is an example of IDS device placement in a generic network. With the IDS and NMS connected to a switch, the switch mirrors incoming traffic to the IDS. The IDS will perform information processing on the forwarded traffic data. If a threat is detected, the IDS will alert the NMS, and the network administrator will take action. If no threat is detected, no alert will be sent, and the network operates as it did before. This operation is applicable for both anomaly and signature types of IDS.

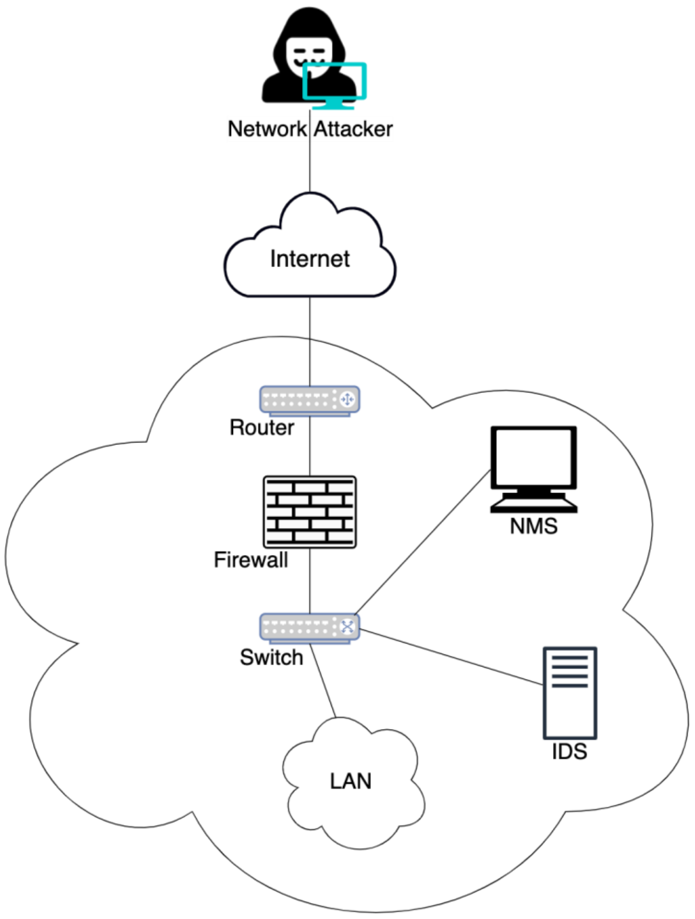


Figure 1 — Generic network architecture with IDS

2.1.2. **Signature Based IDS**

Signature based methods inspect packets by looking at specific data (byte) sequences in traffic payloads. These payload sequences are known to be malicious by cybersecurity researchers. This scheme is similar to anti-virus software's using signature-based methods to determine whether a file on a computer is potentially dangerous. A few key advantages are it offers a high detection rate for known attacks and has fast computation time. However, the downfall is it will only be able to detect known attacks. Zero-day attacks, attacks where their existence is presently unknown by researchers, will not be detected using a signature-based scheme. In addition, the signature database must be frequently maintained and updated. Otherwise, the IDS will miss threats that it easily could've detected.

2.1.3. **Anomaly Based IDS**

Anomaly based methods inspect traffic by using machine learning algorithms. There is a variety of different machine learning algorithms which can be utilized to detect anomalies within networks. This is explored in section 2.4. These models are trained on large sums of benign and/or anomalous network traffic data. The goal of an anomaly IDS is to identify general malicious traffic behaviors within the network, instead of scoping for specific attacks. Anomaly IDS show potential to detect zero-day attacks and attacks which may not have been included in its training process. Given that this type of IDS utilizes a complex algorithm and requires data preprocessing, a major drawback of this IDS for some use cases is the increased computational requirement, and the slower speed compared with signature-based IDS. Another drawback is the introduction of false predications by the IDS. It is a certainty the anomaly IDS will predict false positives (FPs) and false negatives (FNs). Thus, additional security provisions must be used in conjunction to the anomaly IDS.

There are different forms of network data used by different anomaly IDS implementations. The two most common data types are packet based and NetFlow. Most academic IDSs use Netflow, as packet-based inspection is cumbersome. Packet based inspection use a combination of different packet headers as features in the machine learning algorithm. There are thousands of incoming packets per second in a network, and it is extraordinary difficult for a single IDS device to keep up with the throughput in real-time. The utilization of Netflow traffic data is used instead.

2.1.3.1. **NetFlow**

In 1996, Cisco introduced a feature on its routers to simplify analysis of large volumes of packets. Currently, Cisco Netflow version 9 and IETF Internet Protocol Flow Information eXport (IPFIX) are two of the several, most up to date NetFlow standards.

The definition of a flow is best described directly from IETF RFC 3917 [2], "A flow is defined as a set of IP packets passing an observation point in the network during a certain time interval. All packets belonging to a particular flow have a set of common properties". For IDS analysis, utilization of flow-based features sacrifice accuracy for computational speed [3]. Since a flow is a summary of movement of a set of packets, some minor features in a packet which may contain vital attack information are eliminated which may reduce IDS detection. The trade-off is the IDSs ability to analyze all traffic data in real time.

Modern IDS datasets have adopted flow-based features as the standard. Table 1 includes a list and description of some NetFlow version 9 features as an example.

Feature Name	**Description**
IPV4_SRC_ADDR	IP Address of Source Device
L4_SRC_PORT	Layer 4 Source Port
IPV4_DST_ADDR	IP Address of Destination Device
L4_DST_PORT	Layer 4 Destination Port
IN_BYTES	Total Number of Bytes in ingress direction
OUT_BYTES	Total Number of Bytes in egress direction
IN_PKTS	Total Number of Packets in ingress direction
OUT_PKTS	Total Number of Packets in egress direction
FLOW_DURATION	Total duration of flow (in seconds or milliseconds)
TIMESTAMP	System Time when Flow Capture Started

Table 1 — NetFlow features examples

2.2. **Attack Types**

This section describes the general categories of different cyber attacks, and how they are executed.

2.2.1. **DoS & DDoS**

The primary objective of Denial of Service (DoS) and Distributed Denial of Service (DDoS) attacks is to disrupt the ordinary operation of a computer network. These attacks are generally accomplished by targeting the computational resources of key network devices and surrounding network nodes. The mentioned computational resources

consist of; network bandwidth, router's packet forwarding capacity, name servers, memory/computing power on servers, or operating systems' data structures.
DoS attacks manipulate certain functional operations in a system, forces a crash or to overwhelm it. Within the TCP/IP model, these attacks occur on multiple layers, ranging from application, transport, and network layers. An example of a transport layer DoS attacks can range from repeatedly opening an incomplete TCP connection with the victim network to waste its limited capacity of connections. A network layer attack may consist of spamming the victim network with malformed/faulty IP packets to waste the victim's computational resources.

A DDoS is like a DoS attack, but the attacks are executed with more than one host or an army of different hosts to attack a single network. These armies are usually composed of hijacked hosts by the attacker. A commonly used term for this is "Botnets".

2.2.2. **Probe Attacks**

Network probe attacks are used to discover network vulnerabilities, information which will be useful in executing a different type of network attack. There are a variety of different tools that exist to retrieve different network information. These tools can perform; IP scans, port scans, firewall scans and brute force scans for common vulnerabilities for different hosts in a network. Most of such tools are free and open source. Some of these tools are; Nmap, MScan, Security Administrator's Integrated Network Tool (SAINT) and Satan.

2.2.3. **Privilege Escalation Attacks**

Privilege escalation attacks manipulate software or hardware bugs to escalate the attackers' permissions to a superuser, and bypass administrator approval. Once the attacker receives superuser privileges, they have complete control on the software or system they are accessing with elevated permissions and can easily perform malicious actions which may be undetected. An attacker can either upgrade their privilege from a normal user to a superuser, or from a non-existent user to a normal user. Some well-known privilege escalation attacks are buffer overflow attacks, misconfiguration attacks, race-condition attacks, man-in-the-middle attacks, or even social engineering.

2.2.4. **Worm Attacks**

Computer worm attacks are a form of computer virus. Rather than altering a computers filesystem, they attempt to consume computer and network resources or such. Computer worms also take advantage of network vulnerabilities by replicating themselves among other hosts in a network. Several counter measures exist to prevent the spread of a worm such as utilization of anti-virus software, network firewalls and access control lists in networking devices.

2.2.5. **Routing Attacks**

Routing attacks manipulate fundamental functional operations of network routers. There are two type of attacks, open shortest path first (OSPF) protocol attacks, and border gateway protocol (BGP) attacks. OSPF is a protocol used to create a link state table, allowing the router to know the best path to route packets for different hosts. To create this table, the router sends link state advertisements (LSA) messages to survey nearby hosts. Malicious hosts can manipulate various fields in these messages which can lead to an unstable network topology. BGP is a protocol which allows multiple different networks route packets between one another. Routers of different network clusters regularly communicate with each other using BGP updates to ensure standard network operation. Interception and exploits of BGP operations will easily disrupt a single or multiple networks operation. Some examples include black holing (the silent disposal of packets), packet redirection/subversion, both which will lead to network instability.

2.3. **Machine Learning Algorithms**

Machine learning is an evolving research field with a growing domain of applications, including networking, communications, and cyber security. The application of machine learning algorithms is opening a new branch of research in both private sector and academic research. Traditionally, research was executed through analysis, and formulation of mathematical models specific to the research topic to achieve a desired outcome. Machine learning based research operates in a different manner. Instead of a human generating a mathematical model specific to a certain application, machine learning uses a general mathematical form, and tailors' different weights within the model to achieve a certain outcome. This is called “training” the machine learning model, in which the model processes and analyses a dataset and generates a relatively accurate model for the problem at hand. There are several different algorithms which learn to understand problems in different ways.

2.3.1. **Categories of Training**

There are two general categories of machine learning algorithms, each which fits data differently. This section provides a description of these categories.

2.3.1.1. **Supervised Learning**

Most machine learning applications use supervised learning. In supervised learning, data inputs contain a label of desired outcomes. A supervised machine learning model will process the input features and make a prediction. The prediction is then evaluated by comparing it to the labeled output provided in the datasets in the form of a loss function. The model will tune its weights or parameters in a way to achieve the global minima of the loss function. This is an iterative process where in each iteration, the models’ predictions improve, and the loss function consistently decreases and eventually the model will be trained.

There are two main categories of supervised learning problems, classification, and regression problems. In classification problems, the model attempts to predict a predefined/discrete class or category from the input. In regression problems, the model predicts a continuous value based on the input. This is done by shaping a line of best fit to the data.

2.3.1.2. **Unsupervised Learning**

Unsupervised learning models do not use labeled data. The objective of unsupervised learning is to allow the model to independently identify sequences and desired outputs in the data. From a mathematical perspective, unsupervised learning algorithms operate by clustering or dimensionality reduction techniques. Clustering algorithms identify data groups or clusters within a dataset. There are several types of clustering algorithms using various mathematical policies [4], but the most common clustering algorithm is based on kernel methods. Dimensionality reduction algorithms reduce the dimensions of the feature space of a dataset to provide a new projection of a dataset which will clearly highlight any clusters in the dataset. Suppose dataset **X**, of $n \times D$ dimensions where n is the number of data entries, and D represents the number of features in the dataset. The operation of dimensionality reduction will transform dataset **X** into an alternate form, dataset **Y** of d features, where $d < D$ [5]. Further processing of dataset **Y** would indicate clusters of data.

There are a variety of different use cases for unsupervised learning. They are used for anomaly detection, where the unsupervised model is trained to a single category of data and will be able to identify anomalies. Another use case is clustering, to autonomously categorize data based on different densities and clusters of data. The final popular use case is dimensionality reduction, to combine and reduce features to be used by a supervised model to reduce model complexity.

2.3.2. **Machine Learning Algorithms**

There are a multitude of different machine learning algorithms. The mathematics and operation between these algorithms are vastly different. This section explores the inner workings of the chosen machine learning algorithms in this thesis.

2.3.2.1. **Neural Networks**

Neural networks have two modes of operation, feed forward, which is used to classify data, and backpropagation which is used to learn data.

Feed Forward (Classification)

Most neural network types are supervised learning and allow for direct calculation of known outputs from a set of features. Mathematically, the output of a feed-forward neural network is described in a general manner by equation 2.1.

$$\mathrm{y}(x, w) = \mathrm{h}\left(\sum_{\mathrm{j}=1}^{M} w_j \, \phi_j(x)\right) \tag{2.1}$$

Where, *y* is the output vector, *x* is the input features vector and *w* is a vector of weights. Finally, $h(\cdot)$ is a nonlinear activation function and $\phi_j(x)$ is a combination of nonlinear basis functions, which is dependent on previous layers in the neural network.

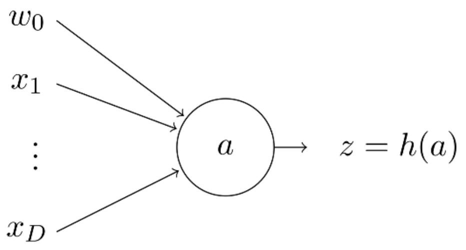


Figure 2 — Activation of a single neuron

The first layer of each neural network consists of *x*. Each following layer of neurons performs its own calculations with the preceding layer of neurons as the input. The output vector before passing through an activation function is called an activation and is denoted by a_j. The activation of the first layer is mathematically described equation 2.2. "D" represents the number of features in the input vector, and (1) indicates the weights corresponding to the first layer. The w_{j0} term is called a bias.

$$a_j = \sum_{i=1}^{D} w_{ji}^{(1)} x_i + w_{j0}^{(1)} \tag{2.2}$$

Once a_j is computed, it is then transformed using an activation function before providing input into the next neuron layer. Equation 2.3 demonstrates this transformation, where z_j is the final output vector for that layer and $h(\cdot)$ is the activation function. Various activation functions exist which may make major impacts on the result of the neural network.

$$z_j = h(a_j) \tag{2.3}$$

The second layer of the neural network may now calculate its outputs. The input to the second layer is z_j. "M" represents the number of neurons in the second layer. The activation of this layer is observed in Equation 2.4.

$$a_k = \sum_{j=1}^{M} w_{kj}^{(2)} z_j + w_{k0}^{(2)} \tag{2.4}$$

A visualization of this neural network can be seen below in Figure 3. Layers in between the input and output are referred to as hidden layers This example demonstrates a single hidden layer neural network. From the activation of the hidden layer, the activation function is applied to receive the final output of the neural network, y_k, as seen in equation 2.5. The dimensionality of y_k is variable on the number of output labels provided when training the neural network. Each value in y_k is correlated with a certain output and contains a probability for classification problems. The output value with the highest probability is rounded up for the neural network's prediction. Neural networks with multiple hidden layers are called deep neural networks (DNN).

$$y_k = \sigma(a_k) \tag{2.5}$$

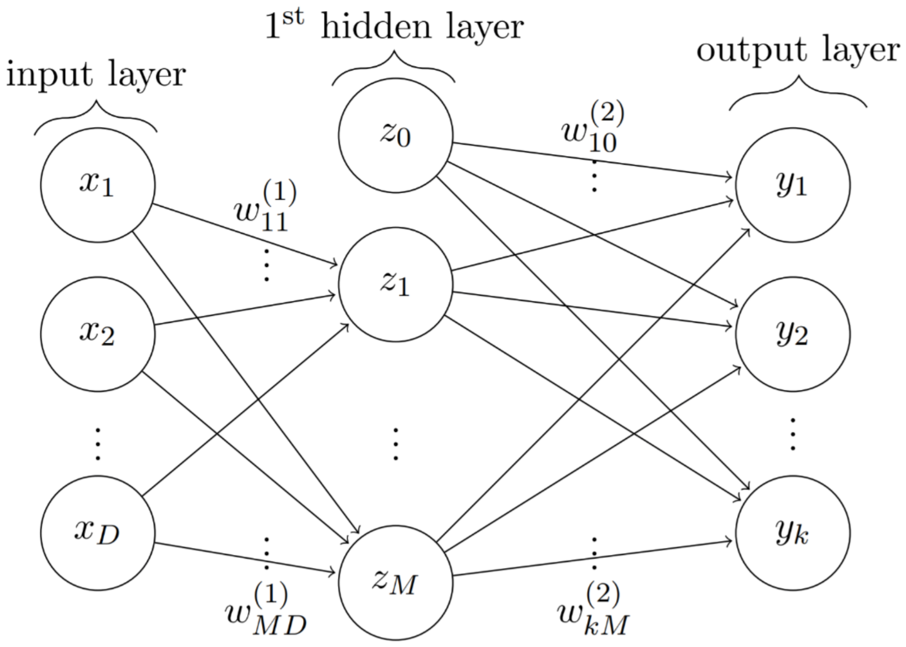


Figure 3 — Visualization of above neural network equations

An activation function decides whether the neuron should be activated and introduces non-linearity into the model. Non-linearity is important because it allows for neural networks to obtain a deep understanding of data by containing multiple layers of neurons. The Rectified Linear Unit (ReLU) is a common activation function because of its simplicity and its handling of negative values. Equation 2.6 describes the ReLU activation function.

$$h(a) = max(0, a) \tag{2.6}$$

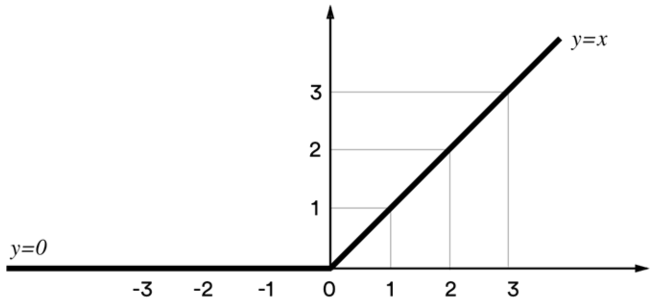


Figure 4 — ReLU graph

For a binary classifier, the output layer contains one node, where the output must represent a probability between 0 and 1. Therefore, a sigmoid activation function is used in the output layer to scale the output of the neural network into a single probability (equation 2.7).

$$h(a) = \frac{1}{1+e^{-a}} \tag{2.7}$$

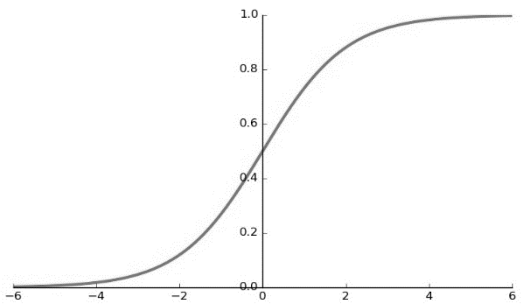


Figure 5 — Sigmoid graph

Backpropagation (Training)

Neural network training is an iterative process, where the neural network will cycle through the entire training data set multiple times. Each iteration of the full training set is called an “epoch”. Within an epoch, the neural network begins training with the feed forward process and the calculation of a loss function. A loss function measures how close a model’s prediction (y) is to the true output (ŷ), thus the goal of a neural network is to minimize the loss function. Loss function minimization is performed using an optimization algorithm such as the Adam optimizer or Standard Gradient Descent (SGD). There are a variety of different loss functions which have specific problem applications. One of the simplest loss functions is the Mean Squared Error (MSE), denoted by equation

2.8, where N is the total samples in the training set. For binary classification, the Binary Cross Entropy (BCE) loss is used (equation 2.9).

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y_i})^2 \tag{2.8}$$

$$\text{BCE} = -\frac{1}{N}\sum_{i=1}^{N}\left(y_i \cdot log(\hat{y_i}) + (1-y_i) \cdot log(1-\hat{y_i})\right) \tag{2.9}$$

Another role of the loss function is to tune the weights within the neural network. Equation 2.10 is an important formula for the backpropagation algorithm to tune the individual weights in the neural network. This equation shows the relationship between the loss function and each individual weight in the neural network. E_n represents the loss function.

$$\frac{\partial E_n}{\partial w_{ij}} = \frac{\partial E_n}{\partial a_{ij}}\frac{\partial a_{ij}}{\partial w_{ij}} = \delta_j z_i \tag{2.10}$$

Where $\delta_j = \frac{\partial E_n}{\partial a_j}$. Upon readjusting the weights in the neural network, the next epoch will commence, where the feed forward propagation is computed with the new weights, and the loss function is calculated for the epoch. Once the loss function is computed, the weights are readjusted, and the next epoch cycle will begin.

2.3.2.2. **Random Forest**

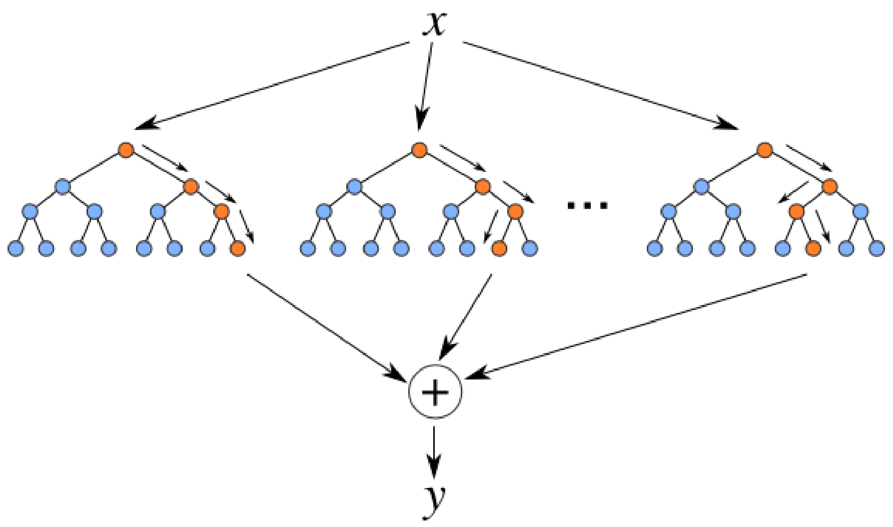


Figure 6 — Visualization of Random Forest voting

A Random Forest (RF) is an ensemble machine learning algorithm. The definition of ensemble learning is a machine learning method that is composed of multiple different learning algorithms that work together to predict the output. RF uses multiple decision trees to make a prediction. Each decision tree within RF is generated during the model training phase. In addition, each decision tree is unique, where while training, a random subset of different features is selected, and the nodes in each tree are built and then split starting from the best features to the worst. A pseudocode algorithm provided by [6] is shown in Algorithm 1. The training set data is denoted by **Z** ($z_1, z_2, \ldots, z_N$) and $z_i = (x_i, y_i)$, and **Z*** are bootstrap samples of **Z** (random sub-samples of data). B denotes the number of bootstrap datasets. T_b denotes a decision tree in the RF. The total number of features is p.

Algorithm 1: *Random Forest*

```
1. For $b = 1$ to $B$:
    a. Generate bootstrap samples, $\mathbf{Z}^*$ of size N from $\mathbf{Z}$.
    b. Grow a random-forest tree $T_b$ to the bootstrapped data. This is achieved by
       recursively repeating steps i to iii (below) for each terminal node of the tree
       until the minimum node size ($n_{min}$) is reached.
          i. Select $m$ variables at random from the $p$ variables.
         ii. Pick the best variables among $m$.
        iii. Split the node into two daughter nodes.
2. Output the ensemble of trees $\{T_b\}_1^B$.
```

The output of a RF for regression problems is described in equation 2.11. The variable x is the input data.

$$\hat{f}_{rf}^{B}(x) = \frac{1}{B}\sum_{b=1}^{B} T_b(x) \tag{2.11}$$

The output of a RF for classification problems is described by equation 2.12. $\hat{C}_b(x)$ is the prediction of the bth RF underlying decision tree.

$$\hat{C}_{rf}^{B}(X) = majority\ vote\{\hat{C}_b(x)\}_1^B \tag{2.12}$$

There are variety of advantages provided by RF. Firstly, it automatically performs feature selection and can determine important features on its own. Secondly, because of the randomness while training, each decision tree is very noisy. This provides two benefits where the RF model is robust to noise in the training data and is unlikely to overfit the data from averaging. Third, the model can handle missing data values. Lastly, RF is efficient on large datasets and can handle many features.

2.3.3. **Evaluation Metrics**

Anomaly detection is binary classification, where there are only two categories of data: anomalous (1) and benign (0). In the machine learning domain, there are metrics used to measure the performance of a binary classification machine learning model. The most important metrics are; recall, precision, F1 score and Area Under Curve (AUC) for imbalanced dataset scenarios.

After the model has been trained, the models prediction results on the test dataset can be analyzed to determine the model's performance via a confusion matrix and its underlying metrics: True Positives (TP), False Positives (FP), True Negatives (TN) and False Negatives (FN). TP/TN are results which the model detected to be/not to be an anomaly respectively which the prediction is correct. FP/FN are results which the model detected to be/not to be an anomaly but the model's prediction is wrong.

Accuracy

The accuracy of a model is the percentage of correct predictions made from the entire dataset. From the four variables, the accuracy is calculated from the formula below. Accuracy is an important metric but cannot be completely relied on to confidently judge a model's performance for unbalanced data. If a dataset contains 90% benign values, and a model predicts 100% of the dataset as benign, then the model will achieve 90% accuracy. Thus the accuracy metric is deceiving and consolation of other metrics is required.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.13}$$

Precision, Recall, F1 Score

Precision, recall and F1 score are analyzed collectively. Precision scores the amount of correct anomaly predictions over the total amount of predicted anomalies. Recall measures the amount of correctly predicted anomalies from the total amount of anomalies in the dataset. Consideration of both these metrics gives a clear view of the model's performance. Another name for recall is sensitivity.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{2.14}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{2.15}$$

The F1 score a combination of precision and recall. It describes overall performance in terms of precision and recall into a single metric.

$$\text{F1} = \frac{2TP}{2TP + FP + FN} \tag{2.16}$$

Specificity

The specificity is also known as the TN Rate (TNR). It is a ratio that measures the rate at which the model correctly detects negative samples over the total of negatives in the dataset.

$$Specificity = \frac{TN}{TN + FP} \tag{2.17}$$

Receiver Operating Characteristic (ROC) & Area Under Curve (AUC)

The ROC graph is an alternate method to judge a machine learning models performance. It shows all confusion matrix values for each classification threshold. The axis of this graph is TP Rate (TPR) vs FP Rate (FPR). For the binary classification scenario, a model will output a probability between 0 and 1. A threshold then must be chosen as a final classification. For example, if the model outputs 0.67 and the threshold is 0.5, the output will be rounded to 1. Typically, 0.5 is the default threshold. The preview of the ROC curve allows for quick understanding of the model's potential performance. The ideal classifier will have a ROC curve that intersects with TPR = 1 and FPR = 0.

The AUC metric is the area under curve. It provides additional detail about the model's performance alongside the ROC curve. Ideally, the AUC should be maximized, where the most optimal AUC value is 1. An AUC of 0.5 is equivalent to that of a random classifier. Anything within range of 0.5 or below means the model performance is equivalent or worse than a random classifier.

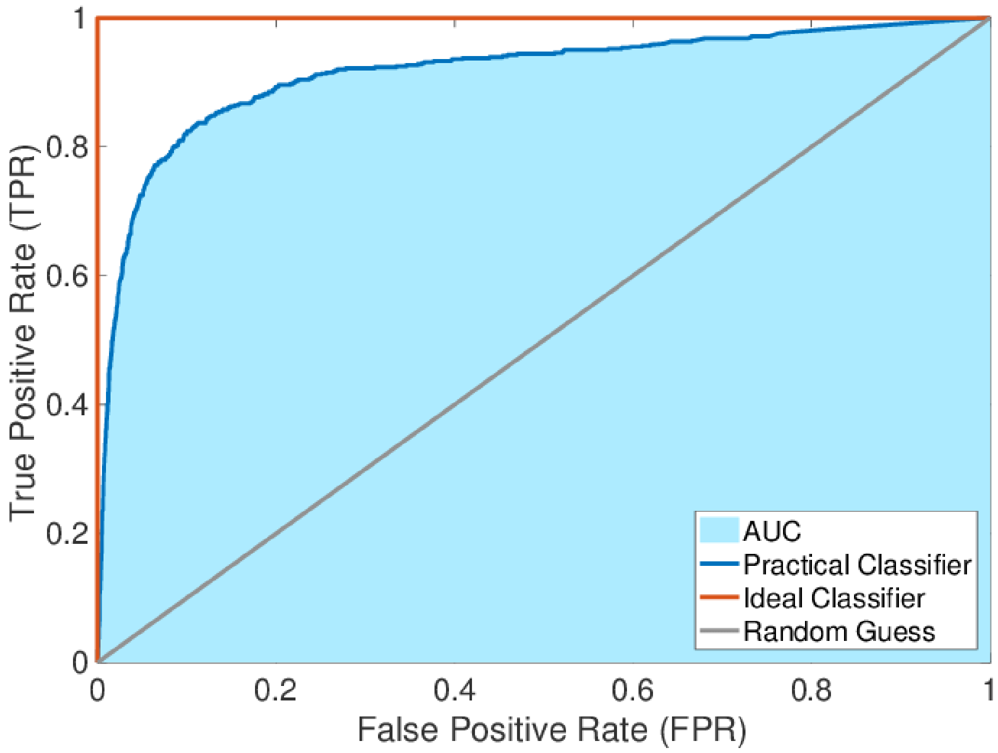


Figure 7 — ROC curves and AUC

A method to discover the optimal classifier threshold from the ROC curve is to use Youden's J statistic, denoted by equation 2.18. This equation is calculated for each point on the ROC curve and determines the optimal threshold.

$$J = \frac{TP}{TP + FN} + \frac{TN}{TN + FP} - 1 \tag{2.18}$$

2.4. Literature Review & Related Works

Several machine learning algorithms may be used to create an anomaly-based IDS. This section will provide an overview of existing research, important metrics to evaluate anomaly-based IDSs and provide background detail for existing IDS datasets.

2.4.1. Overview

In [7], a taxonomy was done analyzing difference implementations of different IDS systems since the conception of an IDS in the early 2000's, until 2020. 97.5% of research relies on a machine learning algorithm, and the remainder is equally distributed between knowledge-based and statistical approaches. Knowledge based systems use finite state machines and rules, and statistical approaches use univariate, multivariate and time series probabilistic models.

From the IDS survey papers [8, 9, 10, 11, 12], the consensus claims deep learning is a better approach for IDS systems for several different network types. Compared to traditional machine learning algorithms, deep learning yields better detection and eliminates the need for feature engineering, allowing for simpler tuning of the model. Thus, in recent academic works, deep learning is the most widely adopted algorithm for anomaly-based IDS [7].

Both supervised and unsupervised deep learning approaches have been implemented. Most models are implemented on the KDD-99 or NSL-KDD dataset. Using supervised learning, deep neural network (DNN), convolutional neural networks (CNN) and recurrent neural networks (RNN) are shown to achieve the best performance. From unsupervised learning, autoencoders (AE), deep belief networks (DBN) and are most widely appreciated.

2.4.2. **Classifier Model Development for IDS**

The scope of this section is to discuss related works where the model is used strictly as a classifier in the IDS pipeline. Works that use multiple datasets do not merge them all into a single IDS like in this thesis, nor do they test their IDS on different datasets.

2.4.2.1. **Random Forest**

[13] uses RF on the NSL-KDD dataset. Their approach uses feature selection to reduce dimensionality and use 10 folder cross validation to train the model. However, the hyperparameters for their decision tree is not specified. Their results claim their model achieves 99% accuracy and detection rate on all attacks. [14] compares support vector machine (SVM) and RF on the KDD ’99 dataset. Their comparison claims SVM produces more accurate results, but RF takes a quarter of the time to train compared to SVM. [15] uses a two-stage approach on the ISCX 2012 dataset. A text-based CNN is

used to extract payload data its output alongside other statistical features is fed into a RF to classify the data. This model achieved 99.13% accuracy.

2.4.2.2. **Deep Neural Network**

A variety of neural network architectures have been tested and implemented on most available datasets. In [16], a distributed IDS was developed using DNN's. The final DNN architecture was developed through rigorous testing of different architectures on five different flow-based benchmark datasets individually for both anomaly and multiclass classifications. This article found a DNN with 5 hidden layers with dropout and batch normalization in addition to feature selection yielded the best results for both anomaly and multiclass classification. Another study [17], developed a custom dataset with flow-based features using the Cooja IoT simulator, and developed a DNN IDS to detect routing attacks in IoT networks. Each dataset focused on a specific attack type. A similar rigorous, trial and error approach was used to test a variety of neural network architectures. The final DNN architecture consists of 5 hidden layers, with neuron counts ranging from 50 to 300 in each layer. Feature selection, dropout and regularization were all used in conjunction with the DNN. The accuracy on their own datasets ranges from 94.5% to 99.5%.

2.4.2.3. **Convolutional Neural Network**

Convolutional Neural Networks (CNN) are mainly used for image processing applications. Unexpectedly within the IDS domain, CNNs have found an appreciation by converting network data into images and passing these images through a developed CNN model. S. Potluri et al. [18] develop a CNN for both the NSL-KDD and UNSW-NB15 datasets by converting each packet into a binary vector dimensions, and then turning each vector in a 8 x 8 grayscale image. The resulting CNN can achieve 91.14% accuracy on NSL-KDD and 94.9% on UNSW-NB15. Other formats are embedding dataset data into matrices and using that as input for a CNN. Other use cases for CNNs are to extract and learn valuable features, which is discussed further in section 2.4.3.1.

2.4.2.4. **Recurrent Neural Network**

Recurrent Neural Networks (RNN) are predominately utilized to supersede flow-based detection methods, in favour of packet level data. For instance, [19] develops and compares LSTM (Long Short Term Memory), Conditional Random Fields (CRF) and Transformer models on their own version of the CIC-IDS2017 dataset using time slot-based features created from the original pcap files. The transformer model substantially outperformed the other two models, and methods such as dropout and feature selection were used. Model hyperparameters were tuned through rigorous testing, and hyperparameters from models with the greatest F-score were selected. Hwang et al. [20] develops an LSTM model for IDS packet level inspection. Their algorithm embeds packet header fields into a sentence and performs anomaly-based detection on three datasets separately. The achieved accuracy is 99%, where the detection time per packet is maximum 2 seconds. In terms of flow based RNN IDS, Yin et al. [21] developed such for both binary and multiclass scenarios, achieving 97.04% accuracy. However, this model is purely reliant on the NSL-KDD dataset and is developed to demonstrate the advantage of deep learning over traditional machine learning algorithms, and therefore has little emphasis on feature selection and hyperparameter optimization.

2.4.2.5. **Autoencoder**

For anomaly-based detection, autoencoders have demonstrated favourable results. Hwang et al. [22] developed a CNN-Autoencoder model for speedy IDS. The CNN module is used to learn important features, and the autoencoder module is used for threat classification. Their model was trained in multiple scenarios (training data was benign only or was mixed between benign and malicious) to determine optimal training scenarios. Also, the medium of inspection is packet based, where the first packets in a flow are inspected. The resulting accuracy is claimed to be near 100%. The majority of autoencoder deployments for IDS are used for feature extraction to optimize other models, which is expanded upon in section 2.4.3.1.

2.4.2.6. **Deep Belief Networks**

DBNs are another form of unsupervised deep learning algorithms that is basically a series of stacked Restricted Boltzmann Machines (RBM). Many DBN use cases are for feature extraction, however DBN for threat classification has been implemented. Zhang et al. [23] develop DBN models on the KDD-99 dataset, where each model is optimized specifically to detect a single attack category. To tune the hyperparameters of each model such as hidden layers, hidden unit, etc., a genetic algorithm was used. The detection rates for each of the four categories range from 97.73% to 99.68%. Tian et al. [24] implement their own methods to improve DBM development using probabilistic mass functions and Kullback-Leibler (KL) divergence at each RBM layer of the DBN to optimize the feature extraction in the model. The accuracy for the tested datasets are, NSL-KDD: 96.17% and UNSW-NB15: 86.49%. Rigorous experimentation was used to develop this model.

2.4.3. **IDS Feature Selection Methods**

This literature review section provides examples of different feature selection methods for intrusion detection systems.

2.4.3.1. **Manual Feature Extraction & Selection**

Another method to select key features for an IDS model is through manual selection. The effectiveness of some features like IP address and port numbers remains unclear. Fernandez and Xu [25] perform a case study for these features. They develop a DNN, and compare the model's performance with, and without IP and port. Their results show including the first three octets IP address and port number may improve performance for a DNN IDS.

2.4.3.2. **Learned Feature Extraction & Selection**

There are a variety of works that use deep learning algorithms specifically for feature extraction to assist threat classifiers achieve better results than using raw feature vectors. The primary methods for feature extraction are CNN, Autoencoder and DBN. A 1-dimension (grayscale image) CNN feature extractor is used in [22], where a set number of packet data from a flow is converted into an image. Smaller size packet data is padded with zeroes to make each pixel in an image correlate with a specific feature. Their CNN attempts to generally categorize each flow by related application type, which is specific to the dataset they use. The trailing autoencoder is used to identify benign and malicious traffic.

Autoencoders are most widely used for IDS feature extraction. Autoencoders may be used to initialize DNN weights and structure, as is done in [26]. An autoencoder is trained on the unlabeled benign data to understand critical features. The trained autoencoder structure and weights is then transferred to a feed forward DNN, where then the model is then retrained using supervised learning. B. Zhang et al. [27] stack an autoencoder on top of a binary tree to perform feature extraction and compare autoencoder and Principal Component Analysis (PCA) for feature extraction. Their results show using a stacked autoencoder improves feature extraction and detection rates collectively. Their hybrid classifier also targets to solve class imbalance issues, a common problem faced by all IDS researchers. Hongpo Zhang et al. [28] use an autoencoder for feature selection and a DNN as a classifier. Using an autoencoder, they are able realize the most impactful features, and select 12 of 202 features as input to their classifier. The final accuracy of their model is 98.8% on the UNSW-NB15 dataset. DBNs are another popular option for feature extraction and selection. Hao Zhang et al. [29] couple a DBN for feature extraction and a series of support vector machines (SVM) for real time classification. The DBN first undergoes a pretraining phase where each RBM layer is trained independently to provide better initial weights for the model. Then the DBN is trained using unlabeled data. Their DBN structure can reduce the dataset feature dimensionality, which is crucial for SVM implementations.

Chapter 3 — **Methodology & Description of Work**

3.1. **Overview**

This section introduces the different processing stages of the proposed work. In addition, details about the development environment and libraries used are provided.

3.1.1. **Pipeline**

A generalized, high-level explanation of the model training and testing process is discussed in this section. For supervised learning, there are three fundamental steps to develop and evaluate a working model: data preprocessing, model training, and model testing.

Data preprocessing is always the first stage, and unarguably the most important stage. The purpose of this stage is to prepare the data for model training. First, the IDS datasets used are duplicated and then sanitized to eliminate faulty data which is unusable. This includes all rows that contain NaN or infinite values which are deleted from the dataset. Next, further preprocessing steps are taken such as organizing data classes and subclass distributions for the training set, generating a validation/test set, scaling the data, and saving the scaled training, validation, and test sets to files.

In total, the UQ-NIDS-v2 dataset contains four datasets from different networks, where each dataset is simulated separately on a unique network. The attack data between each dataset is unique and some datasets have common attack subclasses, but no dataset has a complete overlap. Since the goal of this thesis is demonstrate the potential of an IDS trained on data from a multitude of different networks, the training/validation/test data contains three of four datasets (datasets A, B, C), and the evaluation data is the remaining dataset (dataset D). This simulates an IDS being trained on data from unique foreign networks and being used on the target network. The attack data in the evaluation set is always untouched, but the benign data is altered in some scenarios, which will be discussed further in other sections. If scaling is used on the training set, then the evaluation set it scaled also.

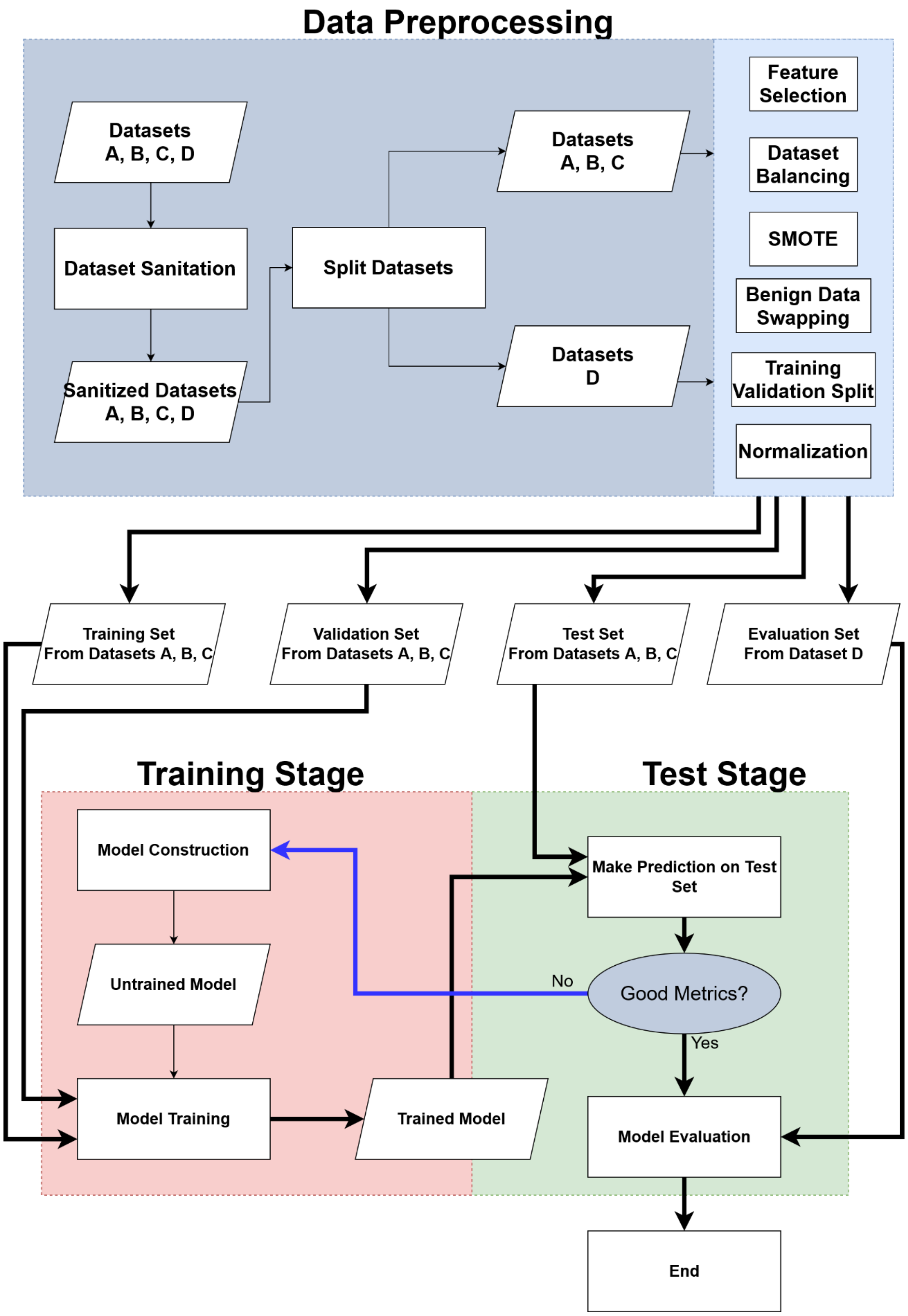


Figure 8 — A flowchart of the general IDS development process

The model training stage is very straightforward. This stage includes choosing a machine learning algorithm, its hyperparameters, architecture and training the model. The training varies by the machine learning algorithm used. For DNN, the model is trained on the training set, and validated using the validation set. For RF, a variety of different models are trained using cross validation, and only the best models proceed to the testing stage.

The testing stage consists of evaluating the model on the test set (data separated from training set) and analyzing the results. Different tools and graphs are used to analyze the model's performance. This includes the ROC curve, the confusion matrix, precision, recall and F1 score. Models that perform well on the training set are then evaluated on the evaluation set. The same tools are used with addition of an attack classification report which graphs the detection for each attack subclass.

3.1.2. **Development Environment**

The work presented is developed using Python 3.8, with combination of a variety of different libraries. The libraries used and their corresponding versions can be seen below in Table 2. The reasoning behind the selection of these libraries is because they are well developed and are widely used by machine learning practitioners, allowing for a smooth and streamlined development of machine learning algorithms. The integrated development environment (IDE) used is JupyterLab version 3.0.14. Pandas is used to manipulate and organize the data. The library used to develop DNNs is Keras TensorFlow and Scikit-Learn for RF. Matplotlib, Scikit-Learn and Seaborn are used to assist with model results analysis.

Library	Version	Description
TensorFlow	2.5.0	Machine learning library with focus on deep learning
Keras	2.4.3	Python interface for simplified implementation of TensorFlow
Matplotlib	3.4.2	Plotting library used for creating plots and graphs
Seaborn	0.11.1	Plotting library with focus on data science
Pandas	1.3.0	Library used for data manipulation and analysis
Numpy	1.19.5	Python interface library to compute mathematical operations
Scikit Learn	0.24.2	Machine learning library with focus on traditional machine learning algorithms, and other machine learning tools
Unbalanced Learn	0.8.0	Library with tools to compensate imbalanced datasets. The primary tool from this library for this thesis is SMOTE.

Table 2 — List of Python Libraries and Packages used

3.2. **Datasets and Data Understanding**

This section provides a detail of the dataset networks, including information such as network types, simulation details, dataset class tallies, included features, and a brief explanation on each attack tools functionality.

3.2.1.1. **Background**

A common issue with existing datasets prior to the creation of the UQ-NIDS-v1/v2 datasets is the lack of overlapping features between different datasets. Because of this, it is difficult to work between different datasets interchangeably and to compare models on each dataset. M. Sarhan et al. [30] from the University of Queensland recompiled multiple popular IDS datasets from their original packet capture (pcap) files to fit all the datasets under the exact same NetFlow features. The four datasets included are the: UNSW-NB15, CSE-CIC-IDS2018, ToN-IoT and BoT-IoT datasets. The UNSW-NB15 and CSE-CIC-IDS2018 datasets are among the most widely used for IDS research. UQ-NIDS-v2 is chosen for several reasons. The shared feature base between multiple datasets allows an IDS to be trained on traffic from a variety of different networks with different applications, and to be evaluated on a totally new network with its own independent purpose and unique attacks. The second reason is because it is reflective of modern networks. Other datasets, like KDD-99 (1999) and its variants receive much criticism because they do not adequately reflect modern networks and applications.

3.2.1.2. **Features**

The names and description of the UQ-NIDS-v2 features is shown. There are 43 total features.

Feature Name	Description
IPV4_SRC_ADDR	IPv4 source address
IPV4_DST_ADDR	IPv4 destination address
L4_SRC_PORT	IPv4 source port number
L4_DST_PORT	IPv4 destination port number
PROTOCOL	IP protocol identifier byte
L7_PROTO	Layer 7 protocol (numeric)
IN_BYTES	Incoming number of bytes
IN_PKTS	Outgoing number of bytes
OUT_BYTES	Incoming number of packets
OUT_PKTS	Outgoing number of packets
TCP_FLAGS	Cumulative of all TCP flags
CLIENT_TCP_FLAGS	Cumulative of all client TCP flags
SERVER_TCP_FLAGS	Cumulative of all server TCP flags
FLOW_DURATION_MILLISECONDS	Flow duration in milliseconds
DURATION_IN	Client to Server stream duration (msec)
DURATION_OUT	Server to Client stream duration (msec)
MIN_TTL	Min flow Time To Live (TTL)
MAX_TTL	Max flow Time to Live (TTL)
LONGEST_FLOW_PKT	Longest packet (bytes) of the flow
SHORTEST_FLOW_PKT	Shortest packet (bytes) of the flow
MIN_IP_PKT_LEN	Len of the smallest flow IP packet observed
MAX_IP_PKT_LEN	Len of the largest flow IP packet observed
SRC_TO_DST_SECOND_BYTES	Src to dst Bytes/sec
DST_TO_SRC_SECOND_BYTES	Dst to src Bytes/sec
RETRANSMITTED_IN_BYTES	# of retransmitted TCP flow bytes (src->dst)
RETRANSMITTED_IN_PKTS	# of retransmitted TCP flow packets (src->dst)
RETRANSMITTED_OUT_BYTES	# of retransmitted TCP flow bytes (dst->src)
RETRANSMITTED_OUT_PKTS	# of retransmitted TCP flow packets (dst->src)

SRC_TO_DST_AVG_THROUGHPUT	Src to dst average thpt (bps)
DST_TO_SRC_AVG_THROUGHPUT	Dst to src average thpt (bps)
NUM_PKTS_UP_TO_128_BYTES	Packets whose IP size <= 128
NUM_PKTS_128_TO_256_BYTES	Packets whose IP size > 128 and <= 256
NUM_PKTS_256_TO_512_BYTES	Packets whose IP size > 256 and <= 512
NUM_PKTS_512_TO_1024_BYTES	Packets whose IP size > 512 and <= 1024
NUM_PKTS_1024_TO_1514_BYTES	Packets whose IP size > 1024 and <= 1514
TCP_WIN_MAX_IN	Max TCP Window (src->dst)
TCP_WIN_MAX_OUT	Max TCP Window (dst->src)
ICMP_TYPE	Type * 256 + ICMP code
ICMP_IPV4_TYPE	ICMP Type
DNS_QUERY_ID	DNS query transaction Id
DNS_QUERY_TYPE	DNS query type (e.g. 1=A)
DNS_TTL_ANSWER	TTL of the first A record (if any)
FTP_COMMAND_RET_CODE	FTP client command return code

Table 3 — List of features and descriptions for UQ-NIS-v2

3.2.1.3. **Data Breakdown**

A breakdown of the UQ-NIDS-v2 data is provided in the table below with counts and description. All the attacks between each dataset are listed under a general category. Clearly, the dataset is very large with 75 million total data samples. Most of the data is anomalous, whereas approximately a third of the entire dataset is benign traffic.

Category	**Count**	**Description**
Benign	25,165,295	Normal unmalicious flows.
DDoS	21,748,351	An attempt like DoS but has multiple different distributed sources.
DoS	17,875,585	An attempt to overload a computer system's resources with the aim of preventing access to or availability of its data.
Probe	6,533,857	A group that intends to collect information about networks ports, applications, IPs.
XSS	24,55,020	Cross-site Scripting is a type of injection in which an attacker uses web applications to send malicious scripts to end-users.
Brute Force	1,274,235	A technique that aims to obtain usernames and password credentials by accessing a list of predefined possibilities.
Injection	687,967	A variety of attacks that supply untrusted inputs that aim to alter the course of execution, with SQL and code injections two of the main ones.

Botnet	143,097	An attack that enables an attacker to remotely control several hijacked computers to perform malicious activities.
Exploits	31,551	Sequences of commands controlling the behaviour of a host through a known vulnerability.
Fuzzers	22,310	An attack in which the attacker sends large amounts of random data which cause a system to crash and aim to discover security vulnerabilities in a system.
Backdoor	18,978	A technique that aims to bypass security mechanisms by replying to specific constructed client applications.
Generic	16,560	A method that targets cryptography and causes a collision with each block-cipher.
MITM	7,723	A method that places an attacker between a victim and host with which the victim is trying to communicate, with the aim of intercepting traffic and communications.
Ransomware	3,425	An attack that encrypts the files stored on a host and asks for compensation in exchange for the decryption technique/key.
Theft	2,431	A group of attacks that aims to obtain sensitive data such as data theft and keylogging
Shellcode	1,427	A malware that penetrates a code to control a victim's host.
Worms	164	Attacks that replicate themselves and spread to other computers.
Grand Total	75,987,976	

Table 4 — List of all attacks, with distributions and descriptions

3.2.2. **Underlying Datasets**

The following section describes the networks used to generate each underlying dataset and provides a tally of different attacks and benign data.

3.2.2.1. **NF-UNSW-NB15**

The original UNSW-NB15 dataset [31] was created in 2015 to address the issues related to the KDD-99 dataset and its variants. The simulation test bed is designed to mimic attacks on a generic corporate network. A traffic simulation tool called IXIA PerfectStorm is used to generate both benign and anomalous data from three different servers to different clients on 2 different networks mimicking attacks from the world wide web. Two of the servers are used to send benign data, and one server is used to send anomalous data. Traffic from the servers to the LANs pass through a firewall device. The attacks are detected and labelled by using a combination of Bro-IDS and Argus IDS.

The UNSW-NB15 version in the UQ-NIDS-v2 (called NF-UNSW-NB15) varies from the original dataset, in terms of amount of data. A distribution of data for the NF-UNSW-NB15 version is provided. The tools used to simulate the attacks are not specified in [31], therefore the mechanics of each attack subclass is unknown.

Class	**Count**
Benign	2,295,222
Fuzzers	22,310
Analysis	2,299
Backdoor	2,169
DoS	5,794
Exploits	31,551
Generic	16,560
Reconnaissance	12,779
Shellcode	1,427
Worms	164

Table 5 — Distribution of attacks in NF-UNSW-NB15

3.2.2.2. **NF-CSE-CIC-IDS2018**

The CSE-CIC-IDS2018 dataset [32] is among the newest datasets available for IDS research. This dataset was generated with major emphasis on representation of modern generic corporate network architectures and modern attacks.
This network is entirely hosted over Amazon Web Services (AWS). Six networks with a total of 420 machines and 30 servers is used to represent a realistic institutional network and one network is used to represent a group of attackers with 50 machines. Each underlying network represents a department, such as information technology, server hosting, research and development, etc. The devices in each network have a variety of different operating systems. The operating systems vary from Windows 8.1, 10 or Windows server 2012, 2016 as well as Ubuntu systems. A custom software was used to generate benign traffic in the network.

The traffic data was extracted using a software application called CICFlowMeter by the Canadian Institute of Cybersecurity at the University of New Brunswick. There are approximately 19 million flows where 88% of the data is benign and the remaining 12%

is anomalous in the original dataset. The distribution of attacks in the UQ-NIDS-v2 version (NF-CSE-CIC-IDS2018) varies from the original. The breakdown of NF-CSE-CIC-IDS2018 attacks, counts tools is provided.

Class	Count	Tool	Target OSI Layer	Target Protocol
Benign	16,635,567	-	-	-
Brute Force	120,912	Patator	7	FTP
				SSH
Botnet	143,097	Ares	7	-
DoS	483,999	Hulk	7	HTTP
		GoldenEye	7	HTTP
		Slowloris	7	HTTP
		Slowhttptest	7	HTTP
DDoS	139,0270	HOIC	7	HTTP
		LOIC	7	HTTP
Infiltration	116,361	Nmap	4	TCP/UDP
Web Attacks	3,502	Custom	7	-

Table 6 — Distribution of attacks in NF-CSE-CIC-IDS2018, description of attack tool

3.2.2.3. **NF-ToN-IoT**

The ToN-IoT dataset (2020) [33] aims to provide an IDS dataset for internet of things (IoT) and industrial IoT (IIoT) networks. The testbed was designed to contain three principal layers: edge, fog and cloud layers. The devices in the edge layer are sensors which collect real world data, VMWare servers, routers, switches, and entertainment devices like smartphones or a smart TV. The fog layer contains a series of different virtualization servers to provide computing capacity physically near the edge layer. These provide different services for the overall network, such as the Node Red service server which generates benign IoT sensor traffic. In this layer, there are 10 hacked Kali Linux systems that run various Bash and Python scripts to exploit vulnerabilities. The cloud layer represents a large size data centre with high computational and storage capacity. Various services in this layer are running such as a website, an IoT hub and HIVE-MQTT (a service management platform for IoT systems). Bro-IDS is used to capture traffic data and generate features. The attacks distribution and tools is provided for NF-ToN-IoT.

Class	Count	Tool	Target OSI Layer	Target Protocol
Benign	6,099,469	-	-	-
Backdoor	16,809	Custom	7	-
DoS	712,609	Custom Python Script using Scapy Library	3	IP
		UFONet	7	HTTP
DDoS	2,026,234	Custom Python Script using Scapy Library	3	IP
		UFONet	7	HTTP
Injection	684,465	Custom	7	-
MITM	7,723	Ettercap	2	ARP
Password	1,153,323	CeWL	7	-
Ransomware	3,425	-	7	-
Scanning	3,781,419	Nmap	4	TCP/UDP
		Nessus	7	-
XSS	2,455,020	XSSer	7	HTTP

Table 7 — Distribution of attacks in NF-ToN-IoT, with attack tools

3.2.2.4. **NF-BoT-IoT**

The BoT-IoT dataset (2019) [34] aims to provide a dataset for scenarios where IoT networks are targeted by Botnet attacks. A Node Red service server is used to connect IoT devices with backend cloud servers. In this testbed setup, a script is run to generate IoT sensor data from temperature, pressure, and humidity sensors. These sensors are used to simulate five different IoT scenarios; a weather station, a smart fridge, motion activated lights, a garage door, and a smart thermostat. These scenarios are hosted on 5 different machines. Benign traffic is generated using the Ostinato tool. The attacks are executed by four different virtual machines (VMs) running Kali Linux and mimic a botnet. The attack distribution of the UQ-NIDS-2 version (NF-BoT-IoT) is provided.

Class	Count	Tool	Target OSI Layer	Target Protocol
Benign	135,037	-	-	-
Reconnaissance	2,620,999	Xprobe2	4	TCP
		Hping3	4	TCP
		nmap	4	TCP/UDP
DDoS	18,331,847	Hping3	4	TCP
		GoldenEye	7	HTTP
DoS	16,673,183	Hping3	4	TCP
		GoldenEye	7	HTTP
Theft	2,431	Metasploit	-	-

Table 8 — Distribution of attacks in NF-BoT-IoT, with description of attack tool

3.2.3. **Attack Tools**

Each individual dataset has a unique suite of tools used to simulate network attacks. This section will discuss functional differences between these tools for different attack types. Discussion of custom or undocumented tools used in these datasets, or any attacks in the NF-UNSW-NB15 dataset is not included. In addition, NetFlow samples from attacks from the NF-CSE-CIC-IDS2018 dataset will be included, as this is the only dataset with labelled tools.

3.2.3.1. **DoS/DDoS**

In each UQ-NIDS-v2 subset, DoS/DDoS compose most of the attacks. In total, there are 8 well known DoS/DDoS tools used between the three datasets. The name of the tools as well as a brief description of how each tools performs, provided in Table 10.

A NetFlow data sample of DoS/DDoS attacks in the NF-CIC-CSE-IDS2018 dataset is shown in Table 9. Notice that except for SlowHTTPTest and Ares, all target ports are port TCP/UDP 80, which is the main designated HTTP port. Each attack operates in a similar manner, where a large volume of requests is sent to a server, and the connection is maintained as long as possible by the attacker. This can be reflected in the IN_PKTS, OUT_PKTS, DURATION_IN, DURATION_OUT, TCP_WIN_MAX_IN and

TCP_WIN_MAX_OUT features. DoS/DDoS attacks of this nature will ideally maximize each of these features.

The only exceptions are LOIC, HOIC and Ares attacks. LOIC will only spam a server with GET requests, and short responses are returned. This can be reflected in the DURATION_OUT feature. HOIC sends and receives data in very short bursts, which is different compared the other DoS/DDoS attacks like HULK, GoldenEye, SlowHTTPTest and SlowLoris. On a data level, Ares appears like LOIC. In conclusion, HOIC, LOIC and Ares are different compared to the other DoS/DDoS tools, and an IDS will not be able to detect this type of attack if it is not included in its training data.

L4_DST_PORT	**IN_PKTS**	**OUT_PKTS**	**DURATION_IN**	**DURATION_OUT**	**TCP_WIN_MAX_IN**	**TCP_WIN_MAX_OUT**	**Tool**
80	27	25	281	234	26883	26847	HULK
80	5	5	3	3	65535	26883	HOIC
80	5	4	32	0	8192	26883	LOIC
80	12	8	890	875	26883	26847	GoldenEye
21	7	7	94	94	26883	0	SlowHTTPTest
80	15	3	16	16	26883	1024	Slowloris
8080	5	5	0	0	8192	26883	Ares

Table 9 — NF-CSE-CIC-IDS2018-v2 data samples for DoS/DDoS attacks

Tool	**Description**
HTTP Unbearable Load King (HULK) [35]	HULK vigorously sends HTTP requests to target web servers to overwhelm them. A HULK flood tries to make the payload pattern in each request unique to evade IDS and IPS devices.
GoldenEye [36]	GoldenEye is a HTTP flooding tool to spam web servers with a large amount of HTTP requests and to keep those connections alive for as long as possible.
Slowloris [37]	Slowloris is a HTTP attack that open multiple connections with a targeted webserver and occasionally sends a partial request header to keep the connections open. Since the targeted webserver is unable to release any of the connections, its resources are consumed and unable to open new connections.
Slowhttptest [38]	SlowHTTPTest is a tool included in Kali Linux, and works in a similar manner to Slowloris, where multiple HTTP connections are opened and are maintained for as long as possible to consume web server resources.
High Orbit Ion Cannon (HOIC) [39]	HOIC is a successor to LOIC, where it spams target servers with junk HTTP GET and POST requests.
Low Orbit Ion Cannon (LOIC) [40]	LOIC is a JavaScript tool that floods servers with either TCP, UDP or HTTP with junk data. The TCP and UDP modes send meaningless message strings to devices at a specified target port, while the HTTP mode spams GET requests.
UFONet [41]	UFONet targets HTTP webservers. It operates as a botnet, where it exploits open redirect vectors on third party web applications to redirect users to the targeted webserver.
Hping3 [42]	Hping3 is a command line tool in Kali Linux to ping devices using either TCP, UDP, ICMP or raw IP protocols.
Ares [43]	A malware run on Android based devices to hijack the devices processing power for botnet purposes.

Table 10 — Known DoS/DDoS tools used in the NF-UQ-NIDS-v2 dataset

3.2.3.2. **Probe**

Probe attacks represent the second largest attack type in each UQ-NIDS-v2 dataset. The most common tool used in the underlying datasets, and in the cyber security industry, for a probe attack is nmap (Network Mapper). Other alternatives of this tool exist but are less commonly used.

Tool	Description
Network Mapper (nmap) [44]	A utility to scan large networks to discover hosts in a network and their associated system information (applications, operating system, packet filters and firewalls, etc.)
Hping3 [42]	Hping3 is a command line tool in Kali Linux to ping devices using either TCP, UDP, ICMP or raw IP protocols.
Nessus [45]	A proprietary software for vulnerability assessment which can detect vulnerabilities such as opportunities for privilege access escalation, default passwords and misconfigurations.
Xprobe2 [46]	An OS fingerprinting tool that relies on fuzzy signature matching, probabilistic guessing, multiple matching, and a signature database.

Table 11 — Known probe tools used in the NF-UQ-NIDS-v2 dataset

The operation of a probe attack from a NetFlow perspective is seen in Table 12. A single packet is sent to a server as ping, and the server will reply. The servers reply will confirm the devices IP address, open/closed ports, applications used, etc. Since datasets using Hping3, Nessus and Xprobe2 do not have the tools labelled with the corresponding data, it is difficult to the differentiate different tools manually.

L4_DST_PORT	IN_PKTS	OUT_PKTS	DURATION_IN	DURATION_OUT	TCP_WIN_MAX_IN	TCP_WIN_MAX_OUT	Tool
53	1	1	0	0	0	0	nmap
5087	1	5	0	0	1024	0	-
25	1	1	0	0	1024	0	-

Table 12 — NF-UQ-NIDS-v2 data samples of probe attacks

3.2.3.3. **Miscellaneous**

The attacks in this section are all the remaining attack tools, which might not be shared in each of the underlying UQ-NIDS-v2 dataset. These attacks and tools represent the minority attack data. None of the datasets explicitly label which data belongs to which tools, thus samples of data for each tool cannot be provided.

Tool	**Attack Type**	**Description**
Patator [47]	BruteForce	A brute force application designed specifically for password guessing. Its able to password guess on a variety of different applications such as FTP, SSH, Telnet, SMTP, SQL, etc.
Ettercap [48]	MITM	A software with a suite of tools to perform MITM attacks.
CeWL [49]	BruteForce	A Ruby application that will generate a list of words from a specified website for Jack the Ripper, a different password cracking software.
Hydra [50]	BruteForce	A parallelized password cracker with a suite of different tools. It can be used on a multitude of different protocols like HTTP, csv, FTP, Cisco AAA, SMPT, SQL, etc.
XSSer [51]	XSS	A framework to detect and exploit XSS vulnerabilities in web applications.
metasploit [52]	Theft	A proprietary cybersecurity tool which can detect vulnerabilities in a network, and hijack a device via command line shell, web application, etc.

Table 13 — Miscellaneous tools used in the NF-UQ-NIDS-v2 dataset.

3.3. **Basic Preprocessing Pipeline**

Before introducing the dataset into a machine learning model, basic preprocessing must be completed. This preprocessing entails sanitizing the data for faulty values like null/not a number (NaN) and infinite values. Rows containing such values are dropped from the dataset immediately. It is also ensured that each row of data has a corresponding correct label and attack category. The sanitized version of each individual dataset is then saved and stored for further preprocessing. A custom preprocessing notebook is developed to generate the train, validation, and testing files. The scope of operation of this custom script includes:

1. Loading data with selected features
2. Balance training datasets to desired benign/anomaly ratios
3. Balance anomaly subclass distributions
4. Split training and validation data (DNN Only)
5. Generate a test set from leftover, unused data
6. Scale training, validation and test sets using Scikit-learn Standard Scaler (zero mean and unit variance scaling)
7. Save datasets to a file

There are multiple reasons for development of a custom preprocessing notebook. First and foremost, the size of the entire NF-UQ-NIDS-v2 dataset is roughly 13 GB. The device being used has 16 GB of RAM, where 5GB is being used to run other applications. The device is not able to load the entire dataset thus, a custom notebook is created to preprocess the data in a computationally efficient manner. Secondly, utilization of a custom preprocessing notebook allows for total control and insight of the data being used. Using the Scikit-learn train_test_split function for instance, does not allow control of selecting anomaly subclass distributions. In addition, it would require the entire dataset to be loaded into the computers RAM.

Regarding the preprocessing pipeline, there are few important details to note. From the four total datasets in NF-UQ-NIDS-v2, one is selected as the final evaluation set (dataset D), and the remaining three (datasets A, B, C) are combined to serve as the training, validation, and test sets. Dataset D represents the target network the IDS is to be implemented on. It’s general network use and attacks contained within the dataset are separate from the other datasets. Thus, the IDS is trained on NetFlow data from foreign networks (datasets A, B, C), and evaluated on the target network (dataset D). A validation set is only generated for DNN and not RF since the RF uses cross validation.

The input to the preprocessing algorithm is the four NF-UQ-NIDS-v2 underlying datasets, A, B, C (used for training, validation, and testing) and dataset D (the final evaluation set). If generating data for a DNN, the output is 12 Numpy files. If generating data for a RF, the output is 10 files.

File Name	Description
X_train.npy	Input data for model training
X_val.npy	Input data for validation during model training (DNN only)
X_test.npy	Input data for testing trained model
y_train.npy	Output labels for model training
y_val.npy	Output labels for validation during model training (DNN only)
y_test.npy	Output labels for testing trained model
X_1.npy	Input data for final evaluation (dataset D)
X_1_ar.numpy	Input data for final evaluation (dataset D), with attacks removed
y_1.npy	Output labels for final evaluation (dataset D), corresponding to X_1.npy
y_1_ar.npy	Output labels for final evaluation (dataset D), corresponding to X_1_ar.npy
a_1.npy	Attack Category Labels, corresponding to X_1.npy and y_1.npy
a_1_ar.npy	Attack Category Labels, corresponding to X_1.npy and y_1.npy

Table 14 — Output files and description of preprocessing pipeline

3.3.1. **The Unbalanced Dataset Problem**

The focus of the section is section is to discuss the preprocessing steps used to address the unbalanced nature of IDS datasets.

3.3.1.1. **Training Set Balancing**

To generate training, validation, and test sets, two primary factors are considered, the ratio of benign data to total data and the distribution of attack subclasses.

Benign Data to Total Dataset (r) Ratio

$$r = \frac{total\ benign\ samples}{total\ dataset\ samples} = \frac{total\ benign\ samples}{total\ benign\ samples + total\ attack\ samples} \qquad (3.1)$$

This metric considers the percentage of benign data relative to the entire dataset. This metric can be described by equation 3.1. This ratio is critical to the performance of the IDS, as it introduces a bias to the model and incorrectly choosing this ratio makes the model prone to the data shift problem. Data shift is the scenario where the distribution of classes between the models' training data and test data varies greatly, which will negatively impact the model's performance on the testing data [53]. In practical scenarios, the ratio of benign to attack data is continuously changing. To address this, an assumption is made to choose this ratio that will reflect an approximate bias in the model. This ratio will likely vary on a case-to-case basis, depending on the model or chosen machine learning algorithm. Some networks may experience more network attacks than others, and some machine learning algorithms may need the bias. The benign samples that are selected are chosen at random.

Attack Subclass Distribution

The distribution of each attack type and tool in the training data play an important role for a successful IDS model. An algorithm is developed to sample attacks in a manner that satisfies the *r* ratio, and the selected attack distribution represents that of the original datasets. The largest attack distributions in each UQ-NIDS-v2 subsets are DoS/DDoS and probe. Therefore, these two attack types are well represented in the training data and are reflective to that of the attack distribution in the final evaluation data. The remaining attacks in the training data are the miscellaneous attacks which are unique to each dataset.

The training, validation set generation process iterates through datasets A, B, C, and they are sampled individually. The sampled attack subclass distributions are proportional to the attack subclass distribution in each of the original training datasets.

3.3.1.2. **Training, Validation, Test Split**

The training set and validation sets are separated using a custom function. The validation set contains the exact same distributions of benign and anomalous subclasses as in the training set. The sampled validation data is removed from the training set.

Once the training and validation sets are generated, a file is created containing all the leftover data not used in the training/validation sets. The same algorithm used to generate the training set is used to create the testing set, but on this leftover data file. The *r* ratio is configurable to increase or decrease the benign data in the test set. The attack subclass distribution reflects the distribution of attacks in the original leftover file.

3.3.2. **Benign Data Training Set Scenarios**

In practical IDS implementations, attack data collection is a constraint. However, this is not the case for benign data, as benign data is abundant because a majority of network traffic is benign. Therefore, the natural benign traffic of the IDS target network can be mixed with the attack data from a foreign network. The developed machine learning models are trained two times on two versions the training data. The training set versions are denoted as: Original Benign Original Anomaly (OBOA) and Replaced Benign Original Anomaly (RBOA). The purpose of these training scenarios is to analyze the IDS understanding of benign data between different networks. A flowchart of how these training sets is generated is provided.

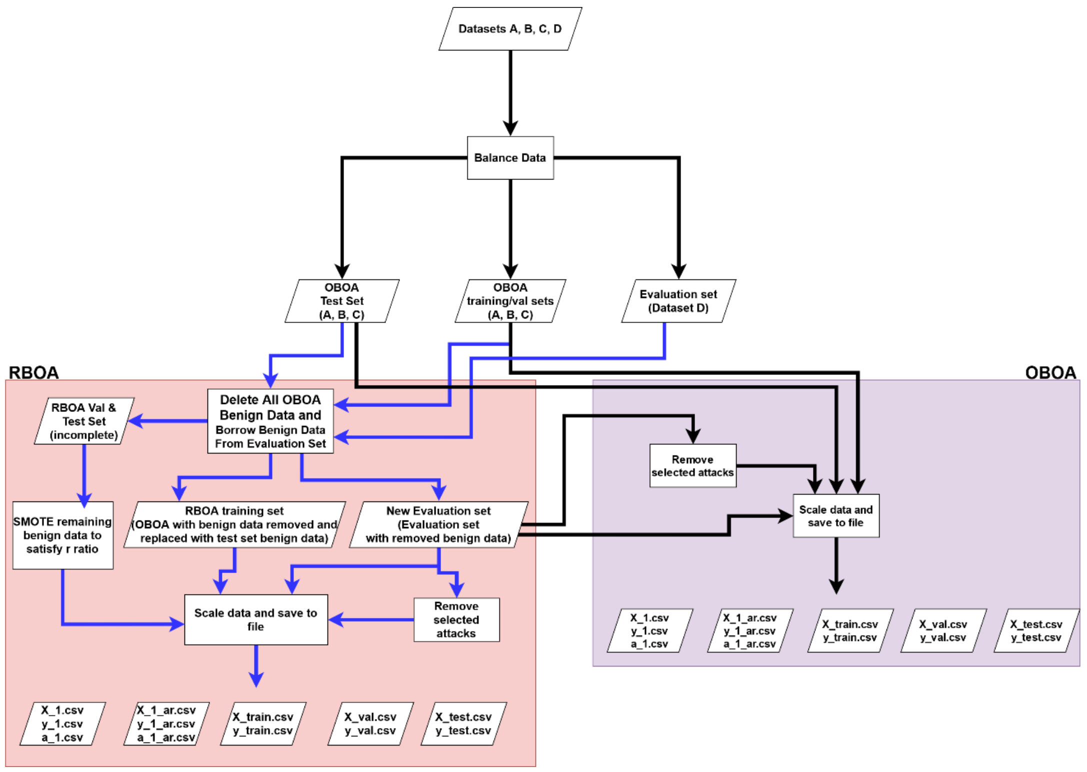

Datasets A, B, C, D
Balance Data
OBOA Test Set (A, B, C)
OBOA training/val sets (A, B, C)
Evaluation set (Dataset D)
RBOA
OBOA
Delete All OBOA Benign Data and Borrow Benign Data From Evaluation Set
RBOA Val & Test Set (incomplete)
SMOTE remaining benign data to satisfy r ratio
RBOA training set (OBOA with benign data removed and replaced with test set benign data)
New Evaluation set (Evaluation set with removed benign data)
Scale data and save to file
Remove selected attacks
X_1.csv y_1.csv a_1.csv
X_1_ar.csv y_1_ar.csv a_1_ar.csv
X_train.csv y_train.csv
X_val.csv y_val.csv
X_test.csv y_test.csv
Remove selected attacks
Scale data and save to file
X_1.csv y_1.csv a_1.csv
X_1_ar.csv y_1_ar.csv a_1_ar.csv
X_train.csv y_train.csv
X_val.csv y_val.csv
X_test.csv y_test.csv

Figure 9 — Generation of training, validation, test set variants.

The attack data in the training, validation, and test sets between OBOA and RBOA are the exact same, the only difference being the benign data. In addition, the evaluation sets between the two scenarios contain the exact same data. The only difference between the two is their scaling, they are scaled relative to their training sets.

3.3.2.1. **Original Benign Original Anomaly (OBOA)**

The data in the OBOA training set is 100% from foreign networks (datasets A, B, C). The data in this set is a compilation of data from other networks with absolutely no connection to that of the evaluation set (dataset D). The configuration, use and applications of the IDS target network is almost entirely different than the data from a foreign network.

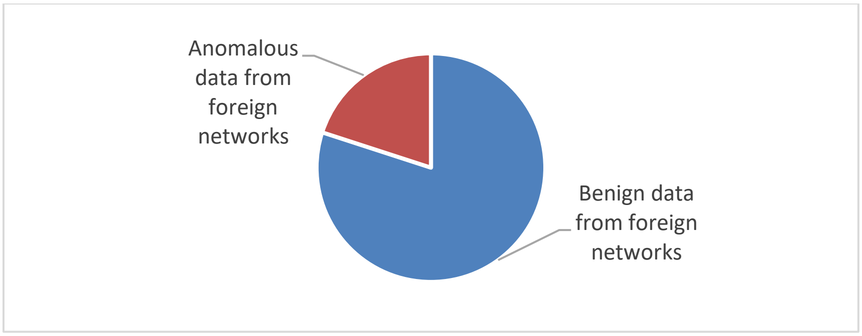


Figure 10 — Example of distribution of benign data for OBOA training set

3.3.2.2. **Replaced Benign Original Anomaly (RBOA)**

The benign data in the RBOA training set is 100% from the IDS target network – the network the IDS is to be implemented on. The benign data is borrowed and removed from the evaluation set. Removal is to ensure no identical duplicates exist between training and evaluation data. Ultimately, the contents of this training set are composed of the benign data of the network which the IDS is to be implemented on, and the anomalous attack data from foreign networks (datasets A, B, C). The attack data is an exact copy of the OBOA set attacks. Only the benign data is removed and replaced. The reason to keeping the exact same attacks is to ensure both versions of the model learn the exact same attacks. The model trained twice on the OBOA and RBOA datasets then may be fairly compared. The validation and test sets also a borrow a small portion of the final evaluation set data. If there is not enough data left to borrow, SMOTE is used to fulfil the remaining samples.

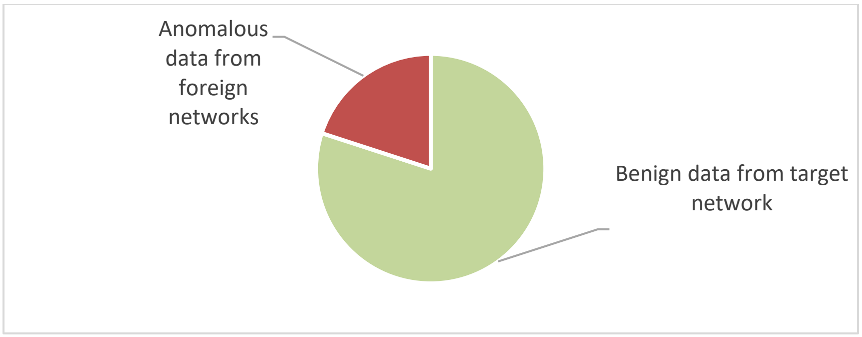


Figure 11 — Example of distribution of benign data for RBOA training set, where the benign data is replaced solely from the test set

3.3.3. **Attacks Removed (AR) Evaluation Dataset**

After the evaluation dataset is scaled and saved to a file, an alternate version of it is processed. This alternate version of the evaluation set is the exact copy of the original evaluation set, except selected attacks are removed. There are two purposes for this evaluation set.

With the attacks removed, the *r* ratio increases, where the distribution of benign data increases. By testing the trained model on this version of the evaluation set, the effect of dataset shift can be analyzed to see the model's performance on a different distribution of benign data. The metric of interest is FPs. This information may be used to tune the model, or to generate a new training dataset with a different *r* ratio.

The distribution of attack subclasses is unbalanced, meaning some attacks have a greater influence on the model's evaluation metrics than others. An attack in the evaluation set which is unique and has zero representation in the training data will most likely be undetected by the trained IDS. For a fair evaluation of the IDS, these attacks are removed for this version of the evaluation dataset.

From a practical perspective, the method of model evaluation may vary on the requirements of the network. For example, if a network contains a signature IDS, and a firewall that both struggle to detect a certain attack, then an anomaly IDS can be trained to detect that certain attack, and the dataset of interest for evaluation would only contain benign data and the attack of interest, like the AR dataset. On the other hand, if an IDS is built to maximize the detection rate on random attacks, even attacks the IDS is not trained on, then the dataset of interest is the full evaluation dataset, containing all attacks.

3.3.4. **Feature Selection**

Feature selection is a critical preprocessing step to have a successful IDS. Many works exist applying different feature selection algorithms and mechanisms on IDS datasets to optimize IDS performance.

In this thesis work, features are selected based on manual analysis. A minimal number of features are selected to reduce model complexity and to hasten training and prediction time. The features chosen are selected based on the following criteria:

1. Features with a noticeably strong correlation with a specific attack. The primary focus for this criterion is DoS/DDoS attacks because of their high volume in the datasets. However, other attacks are additionally considered.
2. Feature representation in the TCP/IP or OSI models. Since different attacks target different protocols on the TCP/IP stack, it is important to have visibility of what is occurring on each layer. Some attacks target specific protocols on different layers. For instance, a DDoS attack spamming Internet Control Message Protocol (ICMP) varies from a DDoS attack vigorously spamming HTTP requests. For the former, inspection of transport layer data and above will not show any obvious hints of a cyber-attack.
3. Features with the probability of randomness or containing arbitrary values are not included. For example, TCP and UDP ports are dynamic/unassigned between the ranges of 49152 and 65535. Most anomalous and benign data in L4_SRC_PORT feature in each dataset is dynamic ports. It is unknown what application is used on the cyber attackers' devices, and therefore, the source port values can be considered arbitrary and not useful.

3.4. **Model Training**

This section focuses on the implementation of the chosen machine learning algorithms, including a brief explanation why the algorithm is selected, and an analysis on the different factors and tools considered in the models training process.

3.4.1. **Deep Neural Networks**

Deep learning offers several advantages for IDS. The nature of the problem explored in this thesis revolves around combining an extremely diverse set of data to train an IDS. Given the large variety of existing attacks, a large volume of data is required to sufficiently cover a wide scope of attacks. Thus, IDSs require a large dataset. Deep learning can understand large, diverse, and complicated data. In addition, deep learning can autonomously perform feature engineering. Lastly, deep learning has a quick prediction time for tabular data, meaning the IDS can scan through traffic in real time. In this thesis, classical DNNs are utilized as opposed to other DNN variants (CNN, RNN, etc.) to minimize preprocessing overhead computation if implemented in a practical environment. Other DNN variants require additional preprocessing steps, such as converting the tabular nature of the data to an image for CNN, or to a sentence format for RNN.

In terms of DNN architecture, several different architectures are experimented and evaluated with, specifically number of hidden layers and the number of neurons in each layer. The activation functions do not change between any tested model. Each model uses ReLU activations in hidden layer neurons, and the output layer is a single node with a sigmoid activation function.

The Adam optimizer is used throughout this entire thesis. Different learning rates are experimented with. If the initial learning rates do not immediately decrease the loss function, then training restarted with a lower learning rate. The chosen loss function is BCE, as it is built to optimize problems of this nature - binary classification. The batch size used in the model varies from model to model.

Two Keras callback tools are used during the model training, Reduce Learning Rate On Plateau (RLROP) and Early Stopping (ES). RLROP actively monitors a specific metric in either the training or validation set through each epoch. Depending on the metric, if it does not improve after a specified number of epochs, the learning rate will be reduced by a specified factor. In this project, RLROP is configured to monitor the validation loss. The factor to decrease the learning rate is 10, and the number of epochs before reducing the learning rate is dependent on the model. Next, the ES callback stops model training when a specified metric does not improve after a specified number of epochs to prevent overfitting and to save time. The number of epochs chosen before stopping is always greater than the specified epochs for RLROP, to prevent the training from stopping before the learning rate is reduced. Typically, the ES epochs is roughly double the RLROP epochs.

In each neural network, a dropout rate of 50% is used in between each layer during training. Dropout drops a specified fraction of weights randomly in a specified layer during training to reduce the chance of overfitting the data. Without dropout, the DNN will overfit the data, thus dropout is always used.

Lastly, the final tool used to assist the neural network during training is class weights. The training data for DNN in this thesis is unbalanced. Because of the unbalanced nature of the training data, class weights are assigned to inform the model of the existing class imbalance, and to allow the minority class to have a bigger impact while training the model. This is achieved by severely penalizing the cost function if it misclassifies the minority class. This is called cost sensitive learning.

3.4.2. **Random Forest**

RF is a suitable machine learning algorithm for IDS use. It can handle large datasets efficiently, automatically handle missing values, and perform automatic feature selection.

The Scikit-learn library *RandomForestClassifer* is used. This RF can perform reasonably well without any configuration, however hyperparameter tuning may be used to improve classifier results. There are five key hyperparameters of interest: number of estimators, maximum number of features, maximum depth, minimum samples split and minimum samples in a leaf.

The number of estimators refers to the number of decision trees to be trained in the random forest. For binary classification problems, the number of trees boosts performance, but this gain is most significant within the first 500 trees. [54]. Performance will keep increasing with more trees, but the overall increase is negligible after 250. Thus, the range of trees is kept from 250 to 500 trees.

The maximum number of features is a hyperparameter that decides the maximum number of features to consider splitting a node. This is the factor that decides the *m* features in Algorithm 1. Scikit-learn offers a variety of different methods to calculate this parameter such as considering the square root of total features or using a logarithm with base 2. This is determined using trial and error.

The maximum depth hyperparameter controls the maximum number of layers within each decision tree. From experimentation, a large maximum depth results in overfitting the dataset. The optimal range for maximum depth must be determined from experimentation.

The minimum samples split specifies the total minimum number of samples to split an internal node inside a decision tree. This hyperparameter coincides with minimum samples in a leaf, where it dictates the minimum number of samples required to generate a leaf node in the decision tree. Both these values are determined experimentally.

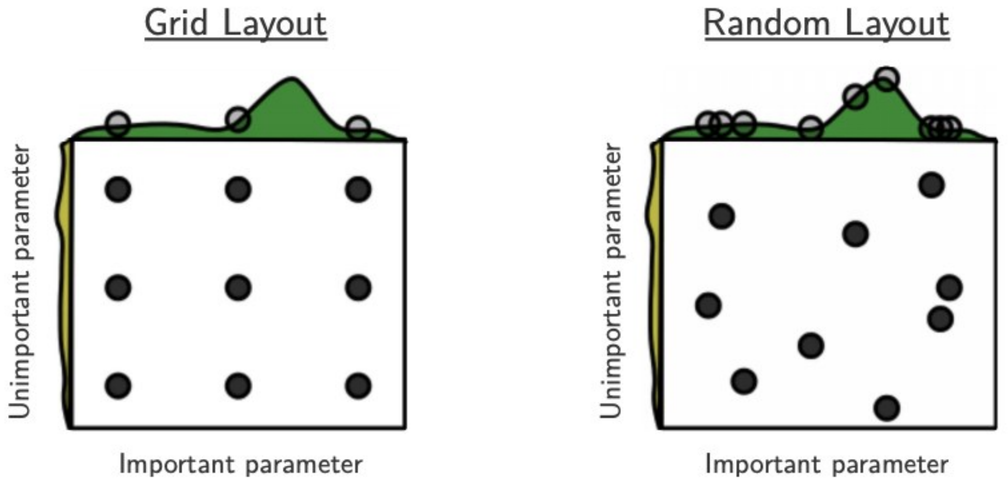


Figure 12 — Grid Search vs Random Search

As many of these hyperparameters are determined experimentally, random search will be used to determine the best models. Random search is method that will test a variety of different models using a random value for each specified hyper parameter via cross validation. An alternative to random search is grid search. Grid search scans through every hyperparameter to determine the optimal model. Grid search can determine the best possible model to use for a problem, but it is extremely computationally expensive and time consuming. Random search is a more efficient method to determine optimal hyperparameters with almost equally good results [55].

3.5. **Evaluation**

Two experiments are used to determine IDS transferability when trained on data from foreign networks. The two experiments use both DNN and RF. The redundancy of repeating these two experiments with different machine learning algorithm re-enforces and verifies the results. In both experiments, the IDS is trained on data from foreign networks, and evaluated on the target network dataset. Each network associated with each dataset is completely independent from each other.

3.5.1. **Experiment 1 – Transferability of Benign and Anomalous Data**

In this experiment, two IDSs are trained, one on the OBOA set and the other on the RBOA set. Both IDS machine learning models use the exact same architecture, hyperparameters, configuration. After training, if the OBOA IDS performs well on the test set, then both IDSs are evaluated on the AR evaluation set. Then the results can be compared, and the impact of training data used can be analyzed.

The transferability of anomalous data is decided by observing the detected attacks in the AR evaluation set for OBOA, RBOA IDSs. If the IDSs can clearly differentiate a large sum of TPs while simultaneously being able to identify a large sum of TNs, then the anomalous data is transferable. More specifically, the recall/sensitivity and specificity must be adequate. The recall is consulted to witness whether most attacks are detected, and the specificity is used verify the recall and to ensure that not all predictions are anomalous. Precision is not factor for the transferability of anomalous data, as this involves analysis and optimization for the transferability of benign data.

The transferability of benign data may only be concluded if anomalous data is transferable. If the anomalous data is transferable, then the precision in both OBOA, RBOA scenarios is consulted. By comparing the precision of both OBOA, RBOA IDSs, it will be clear whether an IDS trained on foreign network benign data will yield as good results as an IDS trained on target network benign data. Failure to demonstrate transferability will yield in an extremely low precision, a high volume of false alarms.

As the transferability of anomalous data primarily involves consideration of the recall, and transferability of benign data primarily involves consideration of precision, the F1 score can be consulted for overall transferability/performance, as it summarizes the two metrics in a single number.

3.5.2. **Experiment 2 – IDS for Maximum Attack Coverage**

The purpose of this experiment is to investigate the full potential of an IDS trained on anomalous data from foreign networks. In practical scenarios, it is very likely for an IDS to face attacks which may be outside of the scope of attacks included in the IDSs training data. Therefore, this experiment analyzes methods to stretch the scope of the IDSs attack coverage.

This experiment trains an IDS on the best performing training set (OBOA, RBOA) observed in experiment 1. The IDS is then evaluated on the full evaluation set, which contains all attacks on the target network. The primary method used to stretch the detection scope is by using threshold adjustment, via Youden's J statistic on the ROC curve. Models with a high AUC demonstrate the potential threshold adjustment to be successful. The results of the model evaluating the full test set using the default threshold is compared with the results using the optimal threshold. Models achieving a high AUC, and high F1 score are considered successful for stretching attack coverage. Dissection of detected attacks will primarily focus on the attacks not included in the AR evaluation set. Other methods to stretch attack coverage like feature selection is explored.

Chapter 4 — **Simulation & Results**

4.1. **Evaluation Dataset**

The NF-CSE-CIC-IDS2018 dataset is selected as the target network. The foreign networks for the training sets are NF-ToN-IoT, NF-BoT-IoT and NF-UNSW-NB15. The NF-CSE-CIC-IDS2018 dataset is chosen as the evaluation set because of its abundance of benign data, allowing for minimal use of SMOTE for the RBOA training sets, and the tools used to simulate each attack are clearly labelled, which gives clear indication of which tools/attacks the IDS struggles to detect for interpretability evaluation.

4.1.1. **Overlapping Attacks**

In total, there are five overlapping general attack categories between the training set and the evaluation set. However, the training and evaluation set only share two exact attack tools: Golden Eye for DoS/DDoS attacks and nmap for network probe. The tools for the remaining common attacks, XSS, brute force and SQL Injection, are custom, therefore their similarity unknown.
The AR evaluation set removes attacks using the following tools: HOIC, LOIC and Botnet (Ares). These attacks are removed based on testing dozens of different models on this data. None of the models can identify any of those attacks for either OBOA, RBOA scenarios using all features. The only exception is LOIC, where strict feature selection must be used, or using an adjusted threshold. Thus, the AR evaluation set will only contain attacks which can be detected by the IDS using the default threshold. The removal of these attacks can be verified by manually analyzing the data, as discussed in 3.2.3.1. It is possible these attacks cannot be detected because, they are not transferable due to network configurations, or there is no similarly between those tools and the tools used in the training data.

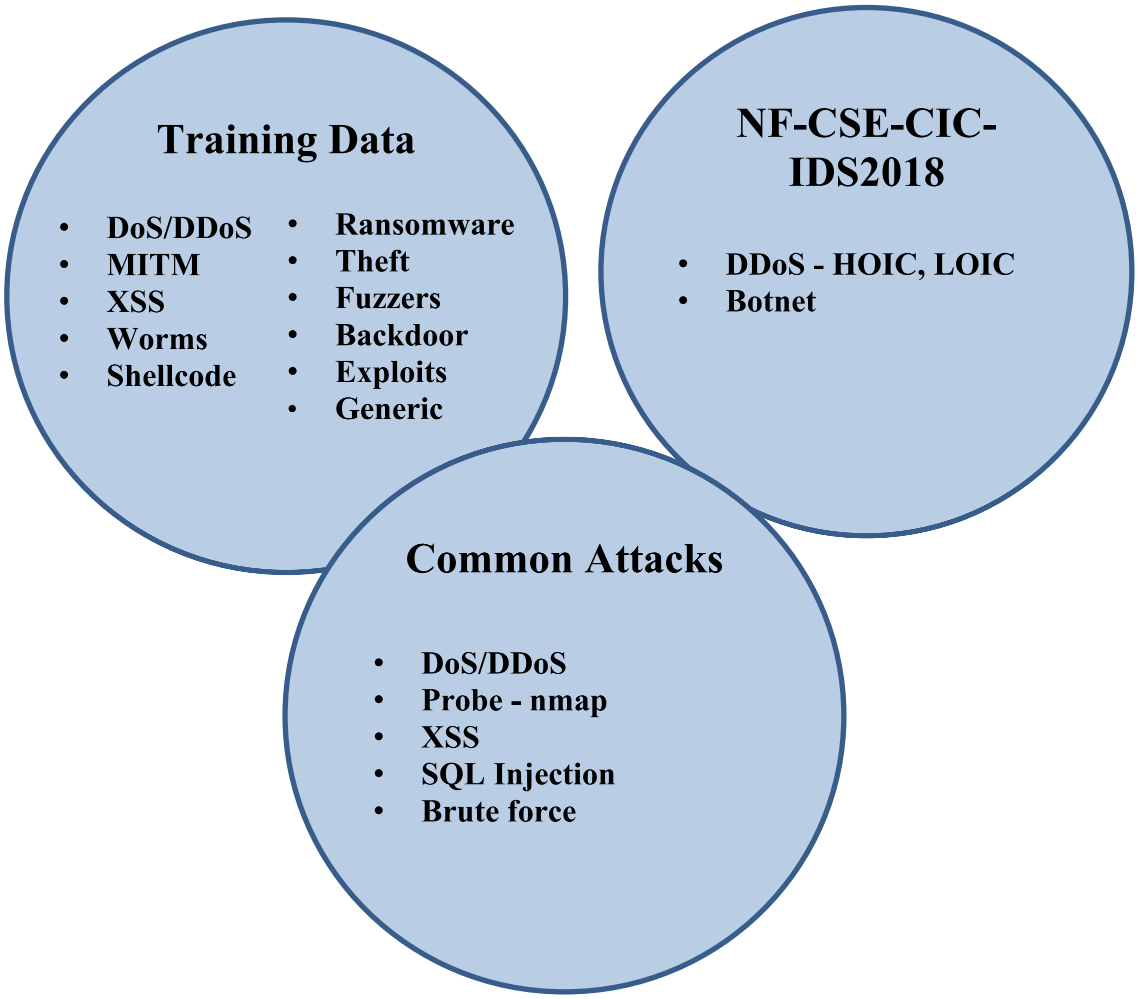


Figure 13 — Estimated overlapping attacks in the NF-CSE-CIC-IDS2018 dataset and training data

4.2. Deep Neural Networks

This section includes all the work done with DNN. First, the details about the preprocessing stage are discussed, followed by results and analysis of both experiments 1 and 2.

4.2.1. Dataset Preprocessing

This section provides tallies/distributions of OBOA, RBOA training, validation, test, and evaluation sets, and lists the selected features with reasoning for selection.

4.2.1.1. **Training, Validation, Test and Evaluation Sets**

The distribution of training data from the foreign networks is seen in Table 15. The main objective in terms of attack distribution in the training set is to include as much of a diverse variety of attacks as possible. The validation data is 12.5% the size of the training data and contains the exact same distribution of benign and attack data. The test data is generated from the remaining unused data after generating the training and validation sets, which utilizes all the remaining unused benign data. The *r* ratio for the training, validation, and test data 80%. The training/validation/test dataset generation algorithm attempts to maximize the available data from each foreign network dataset, depending on if they contain more benign data than attack data, and vice versa. The composition of the OBOA training/validation data from the foreign network datasets is, 92.2% total data from NF-ToN-IoT, 5.7% total data from NF-UNSW-NB15, 2.0% total data from NF-BoT-IoT. Therefore, 95.2% of the training data is from IoT networks. The remaining 5.7% is from a similar network as the target network.

The OBOA and RBOA training sets contain the exact same attack data, and exact same attack category distributions. The only difference between the two is the benign data, where the benign data in OBOA sets stem from the foreign network datasets, and the benign data in the RBOA sets stems from the target network. There is no overlapping/duplicate benign data between the RBOA training, validation, test, and evaluation sets. In addition, around half of the benign data in both the RBOA test and validation sets is generated using SMOTE. SMOTE is performed separately on both sets.

Both evaluation sets for OBOA and RBOA cases are the same and contain the exact same data. The only difference between the two is the scaling. The OBOA evaluation sets are scaled to the OBOA training data, and the RBOA evaluation sets are scaled to the RBOA training data. The original NF-CSE-CIC-IDS2018 contains 16,635,567 benign samples, but the evaluation sets contain 10,020,849 total benign samples. The removed benign samples are used for the RBOA sets. Keeping the number of benign samples constant between OBOA and RBOA evaluation sets allows for clear and equal comparison between scenarios.

	Training Data		Validation Data		Test Data	
Attack	**Count**	**%**	**Count**	**%**	**Count**	**%**
Probe	480728	6.6%	68584	6.6%	59888	2.5%
Backdoor	3969	0.1%	566	0.1%	147	0.0%
Benign	5787878	80.0%	826840	80.0%	1915010	80.0%
DoS/DDoS	369499	5.1%	52938	5.1%	380945	15.9%
Exploits	27637	0.4%	3914	0.4%	0	0.0%
Fuzzers	19452	0.3%	2858	0.3%	0	0.0%
Generic	14465	0.2%	2095	0.2%	0	0.0%
Shellcode	1241	0.0%	186	0.0%	0	0.0%
Worms	144	0.0%	20	0.0%	0	0.0%
Injection	84169	1.2%	12106	1.2%	6003	0.3%
MITM	951	0.0%	135	0.0%	68	0.0%
Brute Force	141837	2.0%	20386	2.0%	10115	0.4%
Ransomware	415	0.0%	67	0.0%	30	0.0%
XSS	302461	4.2%	42855	4.1%	21531	0.9%
Sum	7,234,846	100%	1,033,550	100%	2,393,737	100%

Table 15 — Distribution of training, validation, and test data for OBOA and RBOA sets

	Full Evaluation		AR Evaluation	
Attack	**Count**	**%**	**Count**	**%**
Benign	10,020,849	81.6%	10,020,849	93.3%
Bot	143,097	1.2%	0	0.0%
Brute Force -Web	2,143	0.0%	2,143	0.0%
Brute Force -XSS	927	0.0%	927	0.0%
DDOS attack-HOIC	1,080,858	8.8%	0	0.0%
DDOS attack-LOIC-UDP	2,112	0.0%	0	0.0%
DDoS attacks-LOIC-HTTP	307,300	2.5%	0	0.0%
DoS attacks-GoldenEye	27,723	0.2%	27,723	0.3%
DoS attacks-Hulk	432,648	3.5%	432,648	4.0%
DoS attacks-SlowHTTPTest	14,116	0.1%	14,116	0.1%
DoS attacks-Slowloris	9,512	0.1%	9,512	0.1%
FTP-BruteForce	25,933	0.2%	25,933	0.2%
Infiltration	116,361	0.9%	116,361	1.1%
SQL Injection	432	0.0%	432	0.0%
SSH-Bruteforce	94,979	0.8%	94,979	0.9%
Sum	12,278,990	100.0%	10,745,623	100.0%

Table 16 — Distribution of evaluation sets for OBOA and RBOA scenarios

4.2.1.2. **Feature Selection**

For deep learning, an intuitive approach is used for feature selection. The features are selected with an emphasis based on detected DoS/DDoS attacks as they represent most attacks in the evaluation dataset. An explanation for each selected feature is provided.

Feature	**Explanation**
L4_DST_PORT	Provides insight on Transport Layer activities.
PROTOCOL	Provides insight on Network Layer activities.
L7_PROTOCOL	Provides insight on Application Layer activities.
IN_BYTES	Demonstrates how much data is being sent to the network.
FLOW_DURATION_MILLISECONDS	Demonstrates how long the connection is being kept alive in total. Most DoS/DDoS attacks will maximize this feature.
DURATION_IN	Demonstrates the time being spent for the server to receive data from client. All DoS/DDoS attacks maximize this feature.
TCP_WIN_MAX_IN	Maximum size of TCP window to server. Most DoS/DDoS attacks maximize this feature to keep the connection alive for as long as possible.
TCP_WIN_MAX_OUT	Maximum size of TCP window. DoS/DDoS attacks and some probe attacks maximize this feature.

Table 17— Selected Features with explanation

A variety of other potentially useful features may also be included such as OUT_BYTES, OUT_DURATION, IN_PKTS, OUT_PKTS. The features describing the egress direction flows are omitted to identify LOIC attacks. The LOIC tool is unique, where a minimal reply is sent from the server. None of the training data contains a DoS/DDoS attack of this nature. By omitting this information from the IDS, LOIC attacks appear as attacks in the training data to the IDS.

4.2.2. **Experiment 1 – Transferability of Benign and Anomalous Data**

The contents of this section dictate the architectures and hyperparameters of selected models, and the simulation results alongside key takeaways.

4.2.2.1. **Models**

A summary of models and their hyper parameters is provided below. The models under the model architecture column represent the hidden layers of the neural networks. Each neural network uses the same tools such as RLROP, ES, class weights. Each neural network uses the Adam optimizer, but the learning rates vary. The class weights are consistent in each model, given by the Python dictionary {0:1, 1:5}. This dictionary indicates a 1:5 ratio of anomalous to benign data to the model. RLROP decreases the learning rate by a factor of 10 after the validation loss does not reach a new minimum within a specified number of epochs from the last validation loss minimum. If there is absolutely no improvement after a specified number of epochs from the last validation minimum, ES will stop training all together. Lastly, dropout of 50% is used in each layer for each model to avoid overfitting. No threshold adjustment is used in this experiment, meaning that the default classification threshold of 0.5 is used.

In total, 10 models are trained, where two of each model in Table 18 is trained. One version is trained on the OBOA training set, and the other is trained on the RBOA training set. The results of each are evaluated on their corresponding OBOA or RBOA test set, and on the AR evaluation sets.

Model Architecture	**Learning Rate**	**Batch Size**	**RLROP Epochs**	**ES Epochs**	**Trainable Parameters**
15_5_15_15_20_15	1e-4	64	5	9	1196
20_15_15_8	1e-4	256	5	10	872
20_15_15_15_8	1e-4	256	5	10	1112
60_30	1e-3	128	5	10	2401
64_64_64_64	1e-4	128	5	9	13121

Table 18 — Summary of hyperparameters for tested models

4.2.2.2. **Results**

To evaluate the transferability of attack data, the results from the models trained on the OBOA training data is analyzed. Most OBOA models can detect some samples from each category. Each OBOA model can detect a strong majority of DoS/DDoS attacks, as well as brute force attacks over FTP. However, detection of the remaining attacks varies by model. The remaining attacks are brute force attacks via SSH/Web/XSS, SQL injection and probe attacks (infiltration).

Amongst the five models, the average recall is 0.81 and the average specificity is 0.65 on the OBOA AR evaluation set. This demonstrates the average model can interpret 81% of attacks on the target network and has a rough understanding of what traffic is benign. As the model can predict 81% of attacks correctly while not have a severe bias for predicting anomalies (correctly predict of 65% benign samples), it can be concluded that anomalous data from other networks is transferrable for IDS training.

For transferability of benign data, both RBOA and OBOA IDS results are analyzed. For OBOA, the average specificity and precision are 0.65 and 0.42. For RBOA, the average specificity and precision are 0.99 and 0.91. Although the OBOA IDS shows some understanding of benign data, its results are clearly inferior compared to the RBOA IDS. The OBOA IDS can correctly classify 65% of benign data, and yield more FPs than TPs, where 42% of attack predictions are correct. The average RBOA IDS is superior by minimizing false positives and increasing specificity to 99%. The best performing OBOA IDSs both have 99% specificity and 77%, 95% precision, which is more comparable to the average RBOA IDS. However, the RBOA versions of the best

OBOA IDSs perform better. Thus, a benign data from foreign networks is transferable for IDS interpretation, but the IDS performance is heavily reliant on the model's architecture and hyperparameters. Using benign data from the target network relieves the burden on choosing the optimal architecture an hyperparameters and certainly improves IDS performance.

For overall performance, two different architectures stand out, 15_5_15_15_20_15 and 20_15_15_15_8. The OBOA versions of these models have similar performance, but 15_5_15_15_20_15 has better overall performance reflected in the F1 score (0.81 vs 0.77). The recalls are in similar range, but 15_5_15_15_20_15 performs a better job identifying benign data. Interestingly, the ranking between the two swaps when analyzing RBOA version performance. 20_15_15_15_8 has an F1 score of 0.85 and 15_5_15_15_20_15 is 0.82. Both models perform generally better than their OBOA versions. In addition, these models are no longer the best performing RBOA models, and rank third (20_15_15_15) and fourth (15_5_15_15_20_15). The best performing RBOA model is 64_64_64_64 with an F1 score of 0.88. This OBOA version of this model performs poorly with an F1 score of 0.23. This demonstrates the impact of using benign data from the target network versus foreign networks. Only 2 out of 5 OBOA models are feasible for network implementation, whereas all RBOA models can realistically be used.

Since only 20_15_15_15_8 and 15_5_15_15_20_15 both performed well in the OBOA scenario, their attack detections are compared between the scenarios. The attack report for 20_15_15_15_8 is provided (Figure 14), where the OBOA version is on top, and the RBOA version is at the bottom. The RBOA version has a higher recall and precision compared to OBOA. The RBOA version can detect more SSH brute force attacks and slightly more probe attacks, but it detects less DoS Slowloris attacks, far less web brute force attacks, zero XSS brute force attacks and zero SQL injection attacks. The scope of detection for the RBOA version is slightly narrower than the OBOA version. But the detection of some brute force SSH attacks is boosted.

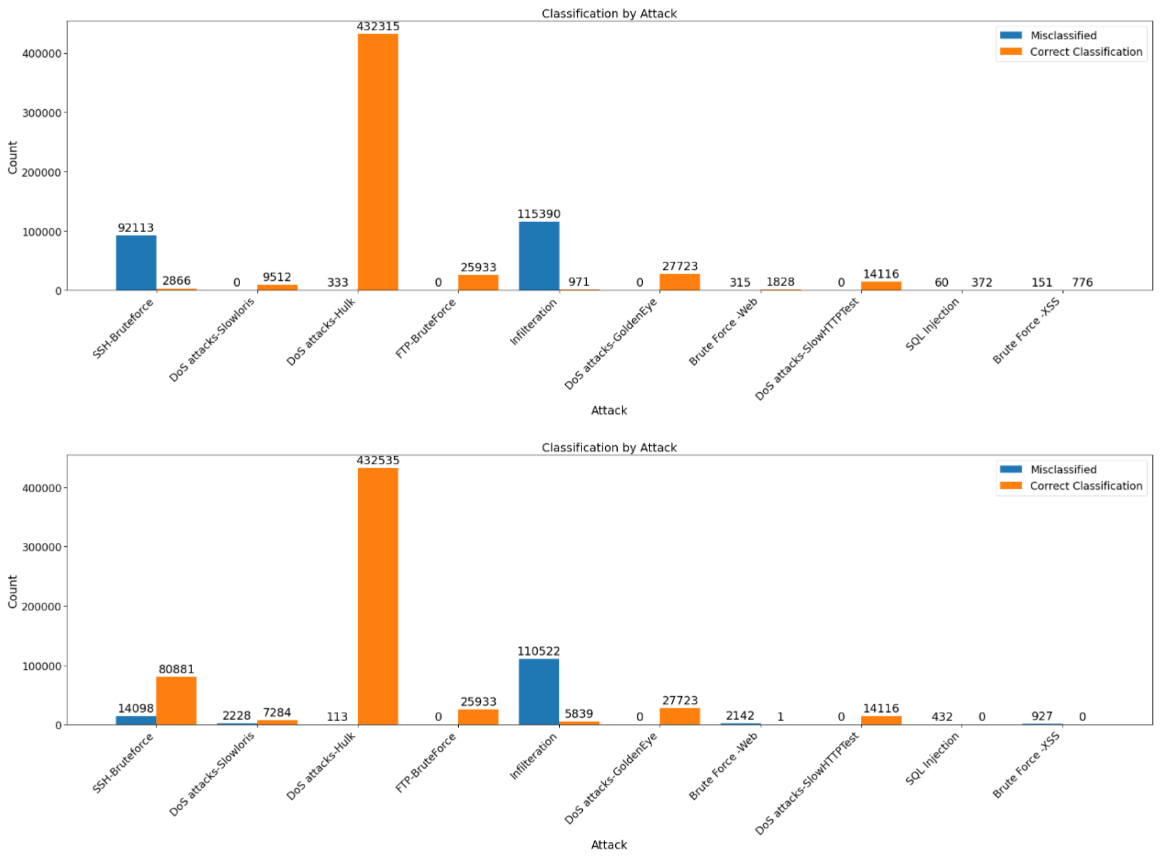
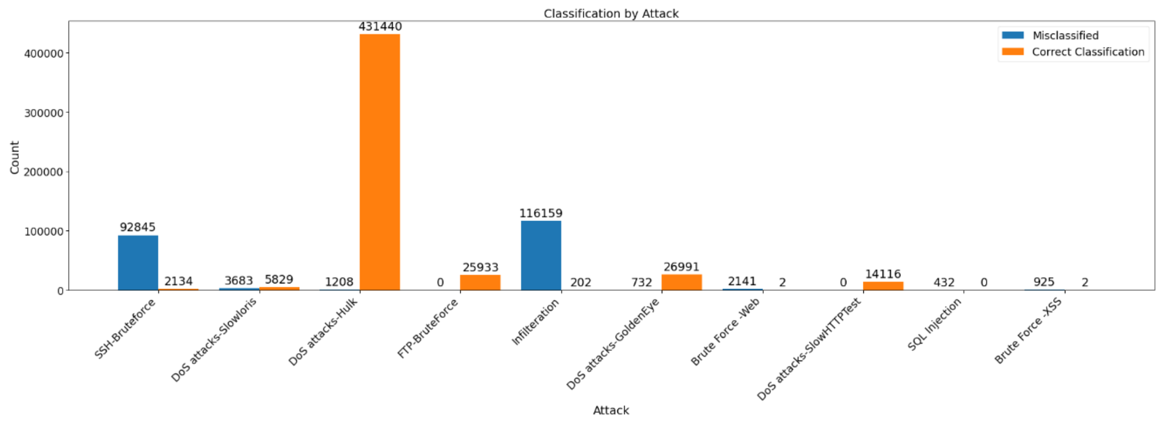


Figure 14 — 20_15_15_15_8 attack report for OBOA (top) and RBOA (bottom) versions

On the other hand, the 15_5_15_15_20_15 attack report (Figure 15) shows no trade-off between OBOA and RBOA versions of the model. The RBOA version is a full improvement of OBOA. The RBOA version can detect all the same attacks as the OBOA version, but slightly more in each category. The only attack not detected by either is SQL injection.



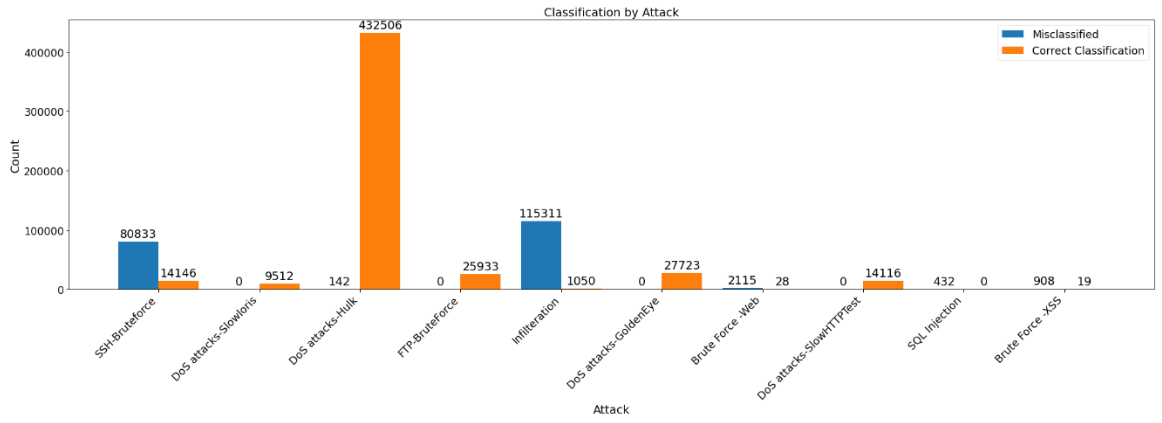


Figure 15 — 15_5_15_15_20_15 attack report OBOA (top) and RBOA (bottom) versions

The difference between a working OBOA model and RBOA counterpart varies by model architecture and hyperparameters. In both OBOA and RBOA scenarios, the IDS model fails to identify an adequate number probe (infiltration) and XSS attacks. The training data and evaluation data both use nmap to simulate probe attacks, and therefore should theoretically be detected by the IDS. Also, the training data contains an ample amount of probe data, with 6.6% total representation. The reason for this failure to detect this attack reflects on the data. The distribution of nmap probe attacks may be underrepresented, where perhaps a majority of attacks in the training data is HPing3 and the two tools operation is not similar. Another reason is it may not be interpretable between networks because of the network configuration. This is a similar case with XSS and injection attacks.

		OBOA		RBOA	
Model	**Metric**	**Test Set**	**Evaluation Set (AR)**	**Test Set**	**Evaluation Set (AR)**
15_5_15_15_20_15	AUC	0.95	0.83	0.94	0.94
	F1 Score	0.82	0.81	0.91	0.82
	Precision	0.95	0.95	0.99	0.96
	Recall	0.72	0.70	0.85	0.72
	Specificity	0.99	0.99	0.99	0.99
20_15_15_8	AUC	0.92	0.84	0.99	0.89
	F1 Score	0.45	0.18	0.94	0.78
	Precision	0.30	0.11	0.96	0.86
	Recall	0.94	0.80	0.92	0.71
	Specificity	0.44	0.51	0.99	0.99
20_15_15_15_8	AUC	0.93	0.89	0.97	0.91
	F1 Score	0.73	0.77	0.95	0.85
	Precision	0.66	0.83	0.97	0.88
	Recall	0.82	0.71	0.94	0.82
	Specificity	0.90	0.99	0.99	0.99
60_30	AUC	0.95	0.83	0.99	0.94
	F1 Score	0.55	0.14	0.95	0.86
	Precision	0.39	0.07	0.97	0.90
	Recall	0.97	0.96	0.93	0.82
	Specificity	0.61	0.14	0.99	0.99
64_64_64_64	AUC	0.95	0.83	0.98	0.94
	F1 Score	0.66	0.23	0.96	0.88
	Precision	0.5	0.13	0.98	0.93
	Recall	0.97	0.88	0.98	0.84
	Specificity	0.76	0.60	0.99	0.99

Table 19 — Summary of results for OBOA and RBOA DNN models

Test Set

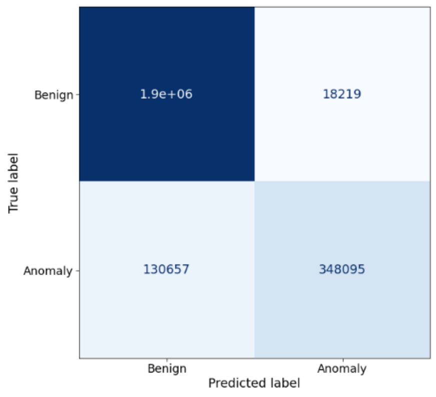
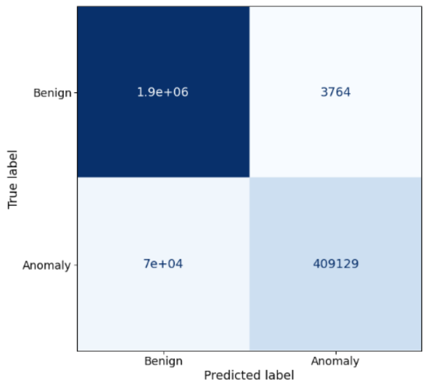
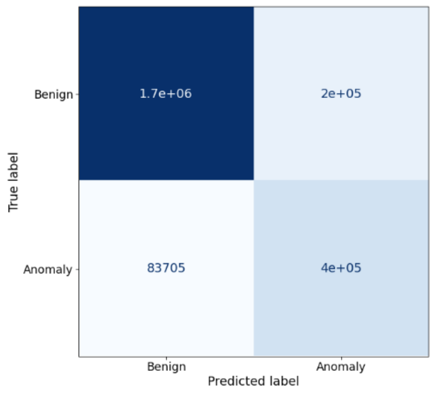
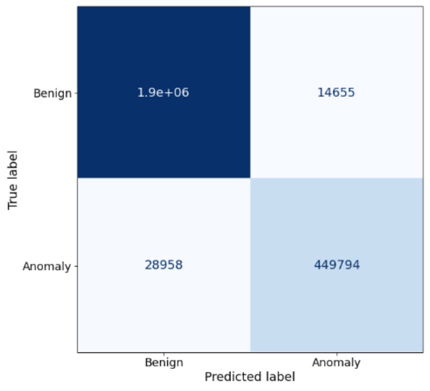
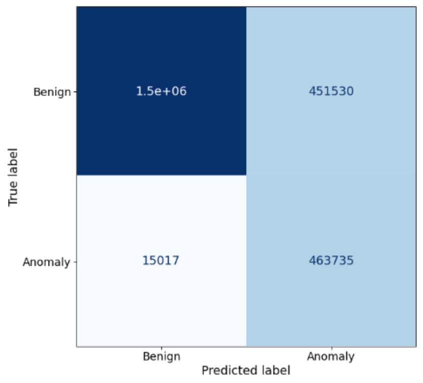
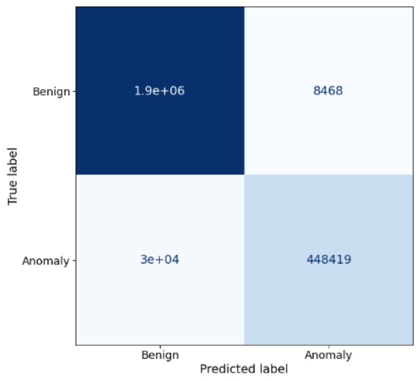
Model	OBOA	RBOA
15_5_15_15_20_15		
20_15_15_15_8		
64_64_64_64		

Table 20 — Confusion matrices for the top three models on the test set

Table 20 shows the confusion matracies for the top three OBOA models on the test set, and the results of the corresponding RBOA models on the RBOA test set. A clear improvement in each model is evident, where for 15_5_15_15_20_15 and 20_15_15_15_8 the overall performance is improved in the RBOA models. For

64_64_64_64, the recall is decreased, but the precision is significantly improved, which is preferred. The other models, 60_30 and 20_15_15_8 also show drastic improvement in the RBOA test set.

AR Evaluation Set

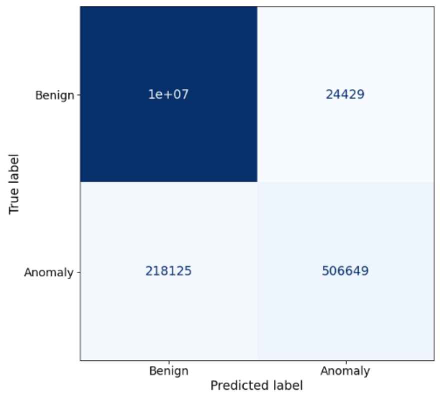
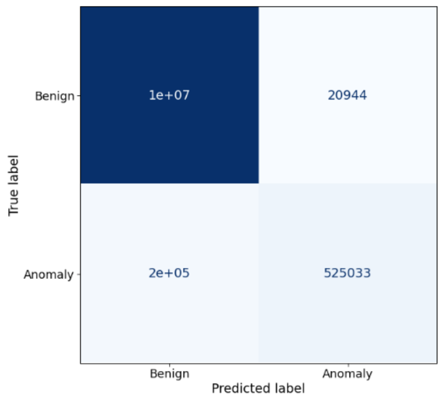
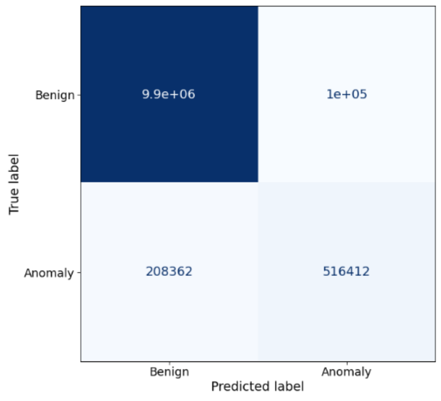
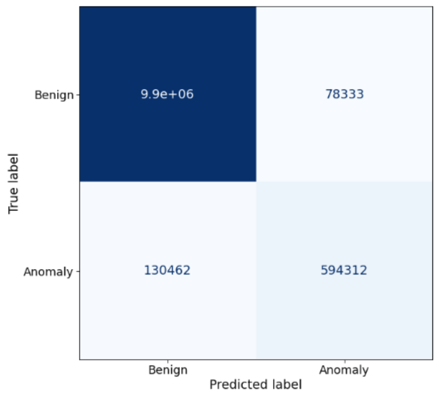
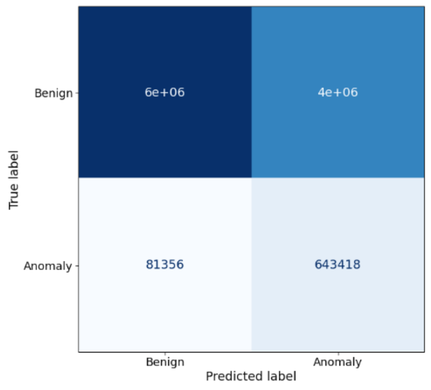
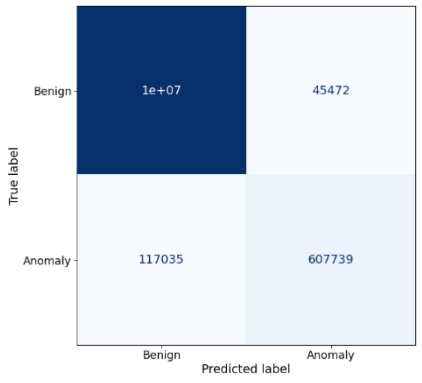
Model	OBOA	RBOA
15_5_15_15 _20_15		
20_15_15_15_8		
64_64_64_64		

Table 21 — Confusion matrices for top three test set models on AR evaluation set

The confusion matrices for the top three OBOA models in the test set phase on the AR evaluation set, alongside their corresponding RBOA models results. OBOA models that perform well on the OBOA test set, have good results on the AR evaluation set. The RBOA version of these models boost the results on the AR evaluation set.

4.2.3. **Experiment 2 – IDS for Maximum Attack Coverage**

In this experiment the neural networks are tasked to detect attacks on the full evaluation dataset. Analysis of how the IDSs reacts to new attacks that are different from the attacks the IDS is trained on can be observed. This dataset contains all the original NF-CSE-CIC-IDS2018 attacks, including LOIC, HOIC and botnet attacks. Because of this new data, it is incredibly challenging to develop an IDS with adequate performance. To overcome this challenge and improve detection, the models are trained on the RBOA training data and are evaluated on the RBOA scaled evaluation datasets. A variety of different neural network architectures are tested but many of those neural networks failed to detect the HOIC, LOIC and botnet attacks. Only the best architectures are included.

4.2.3.1. **Models**

Two architectures with their hyperparameters are shown in Table 22. Both neural networks use the same tools as experiment 1. The Adam optimizer is used with a learning rate of 1e-4. RLROP and ES are used as callbacks, and class weights are used given by the python dictionary {0:1, 1:5}. This dictionary informs the neural network the dataset is imbalanced, where the minority class is 20% of the total training data, so the loss function is penalized heavily when a misclassification occurs during training. Dropout of 50% is used in each layer during training. The loss function used is binary cross entropy.

Model Architecture	**Learning Rate**	**Batch Size**	**RLROP Epochs**	**ES Epochs**	**Trainable Parameters**
15_15_15_15	1e-4	256	5	10	871
15_15_15	1e-4	256	5	10	631

Table 22 — Models and their hyperparameters for maximum detection

4.2.3.2. **Results**

After training a model, the model is evaluated four times, twice on the full evaluation set, and twice on the AR evaluation set, where for each evaluation set, two different thresholds are tested. One threshold is the default threshold, 0.5, and the other threshold is the optimal threshold for the full evaluation set. The optimal threshold is selected using Youdens J statistic from the ROC curve.

The models are selected based on their AUC scores, which showcase the model's potential to perform. For unbalanced problems such as this, the AUC score is a popular metric to use to judge a model's performance. A comparison of ROC curves for the full evaluation set is shown in Figure 16. Based on this graph and the AUC score, it is clear 15_15_15_15 has greater potential to perform on the full evaluation dataset when altering the classification threshold.

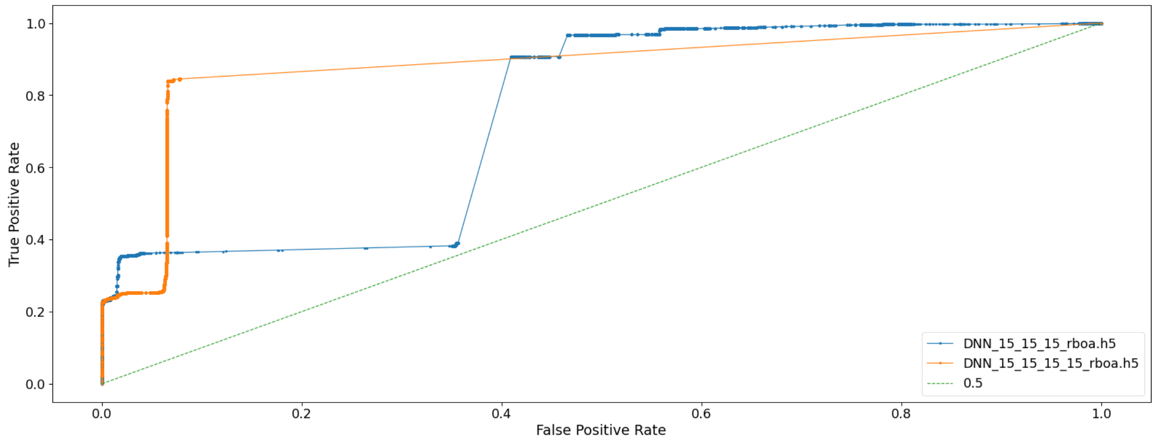


Figure 16 — ROC Curves for models 15_15_15 and 15_15_15_15

Model	Threshold	Metric	Evaluation Set	Evaluation Set AR
15_15_15	-	Optimal Threshold	0.14	0.57
	-	AUC	0.74	0.90
	0.5	F1 Score	0.47	0.66
		Precision	0.68	0.59
		Recall	0.36	0.76
		Specificity	0.96	0.96
		# FPs	384,314	
	0.14	F1 Score	0.46	0.56
		Precision	0.31	0.22
		Recall	0.91	0.95
		Specificity	0.54	0.54
		# FPs	4,584,143	
15_15_15_15	-	Optimal Threshold	0.20	0.56
	-	AUC	0.88	0.87
	0.5	F1 Score	0.36	0.58
		Precision	0.50	0.47
		Recall	0.28	0.76
		Specificity	0.99	0.94
		# FPs	628,887	
	0.20	F1 Score	0.78	0.56
		Precision	0.73	0.44
		Recall	0.84	0.77
		Specificity	0.93	0.93
		# FPs	716,179	

Table 23 — Summary of results for overall classification

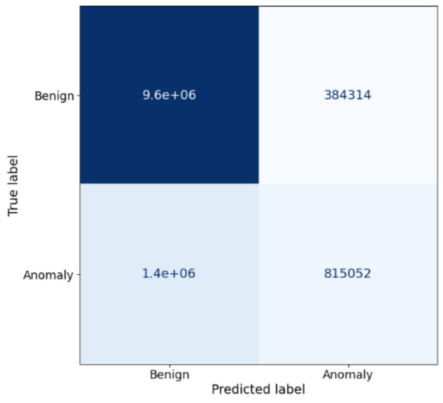
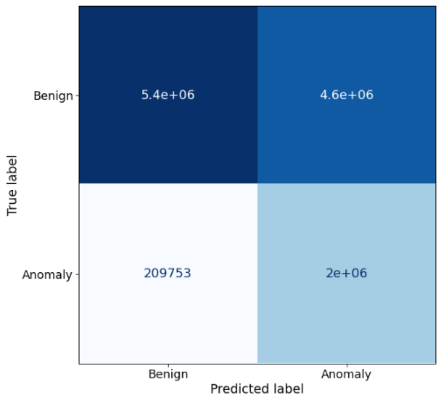
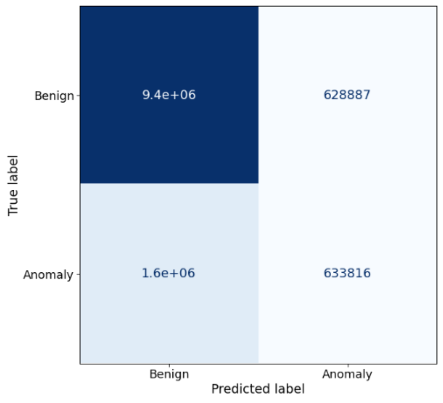
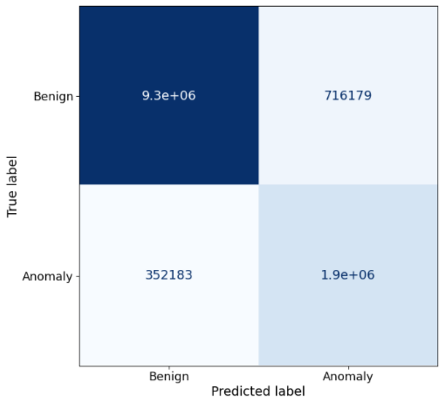
Model	Threshold = 0.5	Threshold = Optimal Threshold
15_15_15 **Optimal Threshold = 0.14**		
15_15_15_15 **Optimal Threshold = 0.2**		

Table 24 — Comparison of DNN model performance with different thresholds.

15_15_15 with the default threshold shows satisfactory performance. It can detect slightly above the expected scope of attacks. The additional detected attacks are LOIC attacks. This is with the assistance of the feature selection scheme. When re-evaluating the model on the full dataset using the optimal threshold, it can detect more attacks. All SSH brute force and HOIC attacks are detected and significantly more LOIC, probe, brute force via web, SQL injection and XSS attacks. However, this is at the expense of a much lower precision, specificity.

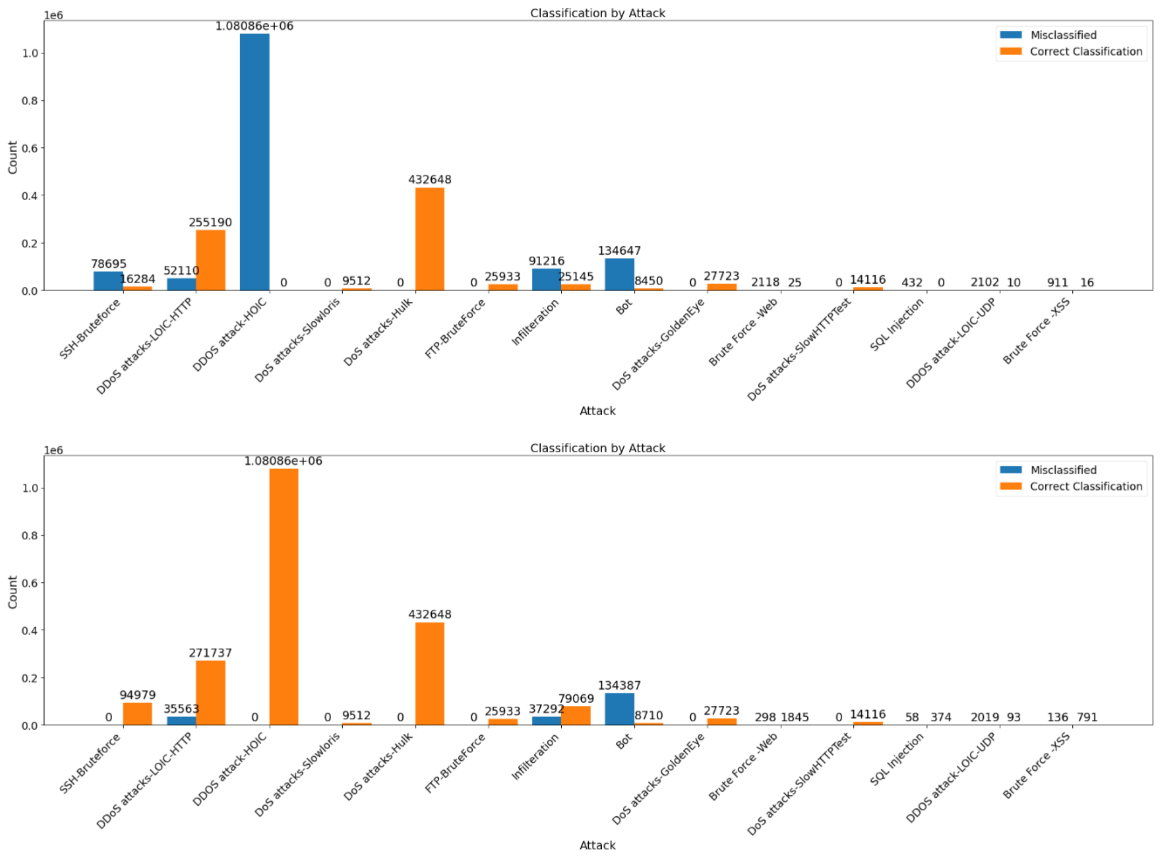


Figure 17 — Attack Classification Report for 15_15_15. Top is threshold of 0.5, bottom is threshold of 0.14

The next tested neural network, 15_15_15_15 does not achieve better results when using the default threshold but is the best performing model using a custom threshold. Using the threshold of 0.5, the model can perform adequately on the expected attacks. It can detect attacks of each category, except for botnet. It can detect a small amount of LOIC, HOIC attacks. Tuning the threshold to 0.2, the model can detect all HOIC and most LOIC, XSS, injection and web brute force attacks. Miniscule detection boost is shown for probe and botnet attacks. Tuning the threshold does slightly penalize the precision and specificity.

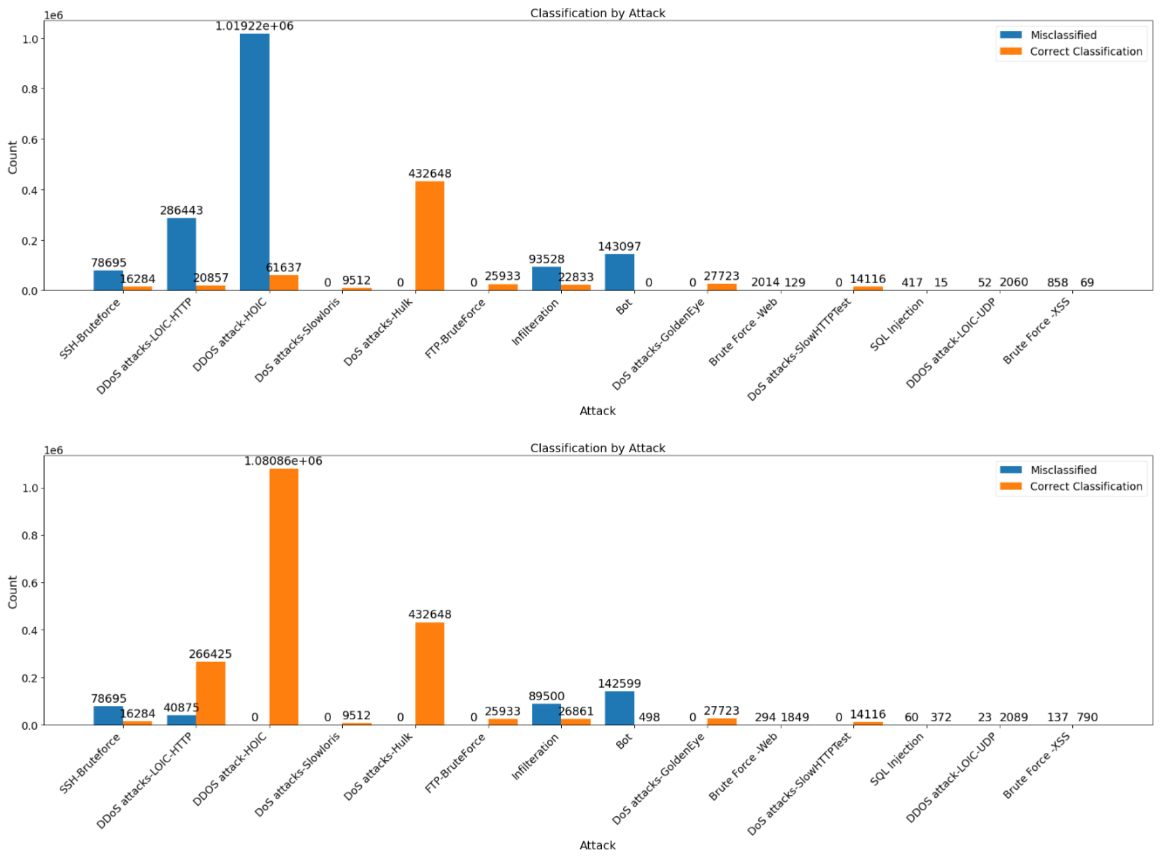


Figure 18 — Attack Classification Report for 15_15_15_15. Top is threshold of 0.5, bottom is threshold of 0.20

Between the models shown in Table 24, the best model to use is dependent on the situation and the goals the IDS intends to achieve. If the goal of the IDS is to detect new attacks, especially DoS/DDoS attacks, using 15_15_15_15 with the threshold of 0.2 is the ideal choice, as it has the best F1 score, maximizing the number of attacks detected, and minimizes the number of false alarms. However, if there are circumstances where the optimal threshold cannot be obtained or cannot be trusted, then 15_15_15 with the default threshold is the next best choice, as it has the highest precision and can detect above the expected scope (LOIC attacks).

4.3. **Random Forest**

This section includes all the work done with RF. First, the details about the preprocessing stage are discussed, followed by results and analysis of both experiments 1 and 2.

4.3.1. **Dataset Preprocessing**

This section provides tallies/distributions of OBOA, RBOA training, validation, test, and evaluation sets, and lists the selected features and reasoning.

4.3.1.1. **Training, Test and Evaluation Sets**

A different package of data is generated for RF. This includes a new training set, test set and evaluation set. No validation set is created because cross validation is used to validate the model's performance during training. The test set is used to verify the model. Finally, the evaluation set is like the DNN evaluation set, except it contains 0.2% extra benign data. None of these datasets are scaled, as scaling is not required for RF. The test set is a 25% split from the training data.

The same method is used to organize the datasets. The total distributions between train and testing sets from each dataset are NF-BoT-IoT composes 4.6% of with 264,632 total samples, NF-ToN-IoT composes 93.4% with 5,404,368 total samples, and NF-UNSW-NB15 composes 2% with 117,354 total samples. The *r* ratio for each is 50%, an equal split between benign and anomalous data.

With regards to the OBOA and RBOA versions, the attack data in each is the exact same, and the total benign samples in both are the exact same. The only difference between the two is the RBOA benign data is from the target network. SMOTE is not used to generate any new samples for the RBOA train and test sets.

	Training Data		Test Data	
Attack	**Count**	**%**	**Count**	**%**
Probe	452,777	5.9%	151,794	5.9%
Backdoor	5,110	0.1%	1,684	0.1%
Benign	3,834,765	50.0%	1,278,255	50.0%
DoS/DDoS	2,116,147	27.6%	704,465	27.6%
Exploits	21,714	0.3%	7,233	0.3%
Fuzzers	11,341	0.1%	3,965	0.2%
Generic	2,246	0.0%	768	0.0%
Shellcode	532	0.0%	179	0.0%
Worms	108	0.0%	35	0.0%
Injection	196,138	2.6%	65,549	2.6%
MITM	2,215	0.0%	694	0.0%
Brute Force	323,834	4.2%	107,763	4.2%
Ransomware	969	0.0%	310	0.0%
XSS	701,517	9.1%	233,783	9.1%
Sum	7,669,413	100%	2,556,477	100%

Table 25 — Distribution of training and test data for OBOA and RBOA sets

	Full Evaluation		AR Evaluation	
Attack	**Count**	**%**	**Count**	**%**
Benign	10,370,746	82.3%	10,370,746	93.5%
Bot	143,097	1.1%	0	0.0%
Brute Force -Web	2,094	0.0%	2094	0.0%
Brute Force -XSS	895	0.0%	895	0.0%
DDOS attack-HOIC	1,080,858	8.6%	0	0.0%
DDoS attacks-LOIC-HTTP	280,337	2.2%	0	0.0%
DoS attacks-GoldenEye	27,723	0.2%	27,723	0.2%
DoS attacks-Hulk	432,487	3.4%	432,487	3.9%
DoS attacks-SlowHTTPTest	14,116	0.1%	14,116	0.1%
DoS attacks-Slowloris	7,227	0.1%	7,227	0.1%
FTP-BruteForce	25,933	0.2%	25,933	0.2%
Infiltration	114,326	0.9%	114,326	1.0%
SQL Injection	432	0.0%	432	0.0%
SSH-Bruteforce	94,979	0.8%	94,979	0.9%
Sum	12,595,250	100.0%	11,090,958	100.0%

Table 26 — Distribution of evaluation sets for OBOA and RBOA versions

4.3.1.2. **Feature Selection**

In total, 27 features are used for the RF model. As RF has automatic feature selection built into its algorithm, there is no real need for manual feature selection unless reducing computational complexity or manipulating features to boost model performance.

By this, this is the same reason OUT_BYTES, OUT_DURATION, IN_PKTS, OUT_PKTS is excluded in the deep learning experiments: to detect LOIC attacks. Prior experimentation with these features made it impossible for models to detect LOIC attacks. However, for experimentation purposes, these features are included to witness the number of trees that are generated without these features, revealed in experiment 2.

Feature Name	Feature Name	Feature Name
L4_DST_PORT	MIN_TTL	TCP_WIN_MAX_IN
PROTOCOL	MAX_TTL	TCP_WIN_MAX_OUT
L7_PROTO	LONGEST_FLOW_PKT	ICMP_TYPE
IN_BYTES	SHORTEST_FLOW_PKT	ICMP_IPV4_TYPE
IN_PKTS	MIN_IP_PKT_LEN	DNS_QUERY_ID
OUT_BYTES	MAX_IP_PKT_LEN	DNS_QUERY_TYPE
OUT_PKTS	DURATION_IN	DNS_TTL_ANSWER
TCP_FLAGS	DURATION_OUT	FLOW_DURATION_MILLISECONDS
CLIENT_TCP_FLAGS	FTP_COMMAND_RET_CODE	SERVER_TCP_FLAGS

Table 27 — Selected 27 Features for RF

4.3.2. **Experiment 1 - Interpretability of Benign and Anomalous Data**

This experiment is an exact repeat of 4.2.2 and verifies the established conclusions. Each selected RF model is trained twice, once of the OBOA dataset and another on the RBOA dataset. Once trained, both OBOA and RBOA versions of the RF are evaluated on the exact same AR evaluation set.

4.3.2.1. **Models**

Five models are chosen from three different random searches using 5-fold cross validation. All random searches are executed on the OBOA training data. E11, and E21 are the best models from two separate random searches. E01, E02, E03, are the top three RFs from the last random search, which tested the most models.

Model	N estimators	Max features	Max depth	Min samples split	Min samples leaf
E01	300	Log2	20	7	3
E02	400	Log2	20	3	7
E03	400	sqrt	20	11	20
E11	500	sqrt	15	8	12
E21	250	sqrt	12	3	15

Table 28 — Random Forest Models

4.3.2.2. **Results**

The results from each OBOA, RBOA model on the test sets demonstrates exceedingly great performance. Each RF model can near perfectly fit all the of the training data and understand each attack within it. The F1 score for each RF model on the test set is either 0.99 or 0.98, indicating a strong reliable classifier. If this IDS is implemented on the same networks it received its training data, then it would perform perfectly for detection of the expected scope of attacks.

However, the goal of this IDS is to be implemented on the NF-CSE-CIC-IDS2018 target network. The average OBOA IDS recall in Table 30 on the AR evaluation set is 0.69, meanwhile the average specificity is 0.93. This indicates the OBOA models can correctly identify 69% of total attacks, and 93% of total benign traffic. This demonstrates the model can roughly distinguish most attacks from benign data, and the model does not classify all traffic as anomalous. Therefore, the transferability of anomalous data is verified with RF using a completely new training procedure and training set.

Comparison of the OBOA, RBOA RF models on the AR evaluation set will verify the transferability of benign data. The average OBOA AR evaluation set precision and specificity is 0.39 and 0.93. For RBOA, the average precision and specificity is 0.99 and 0.98. For the average OBOA IDS, it can predict 93% of benign traffic, but 39% of the anomalous predictions are correct, indicating many false positives. The average RBOA IDS can identify 99% of total benign data and 98% of its anomalous predictions are correct. This demonstrates that an IDS trained on benign data from foreign networks can interpret traffic on the target network, but the IDS will perform better when trained on benign data from the target network.

The overall performance of the OBOA RFs is heavily penalized by the large volume of FP predictions. The average F1 score is 0.50, where the average precision is 0.39 and average recall is 0.69. Although feasible for network implementation, the OBOA IDSs are not a completely trustworthy because of the high volume of false positives. The best performing OBOA IDS is E03, with F1 score of 0.52, and precision, recall of 0.42, 0.70. The RBOA counterparts of each IDS are superior. The average RBOA IDS F1, precision and recall are 0.88, 0.98 and 0.8. In addition to the decreased number of FPs, the RBOA model can predict 11% more attacks using the exact same attack data during training. This verifies the impact the benign data has on the IDS, and the RBOA training set generation procedure should be used. The best performing RBOA IDSs are E03 and E21 both with F1 score of 0.92.

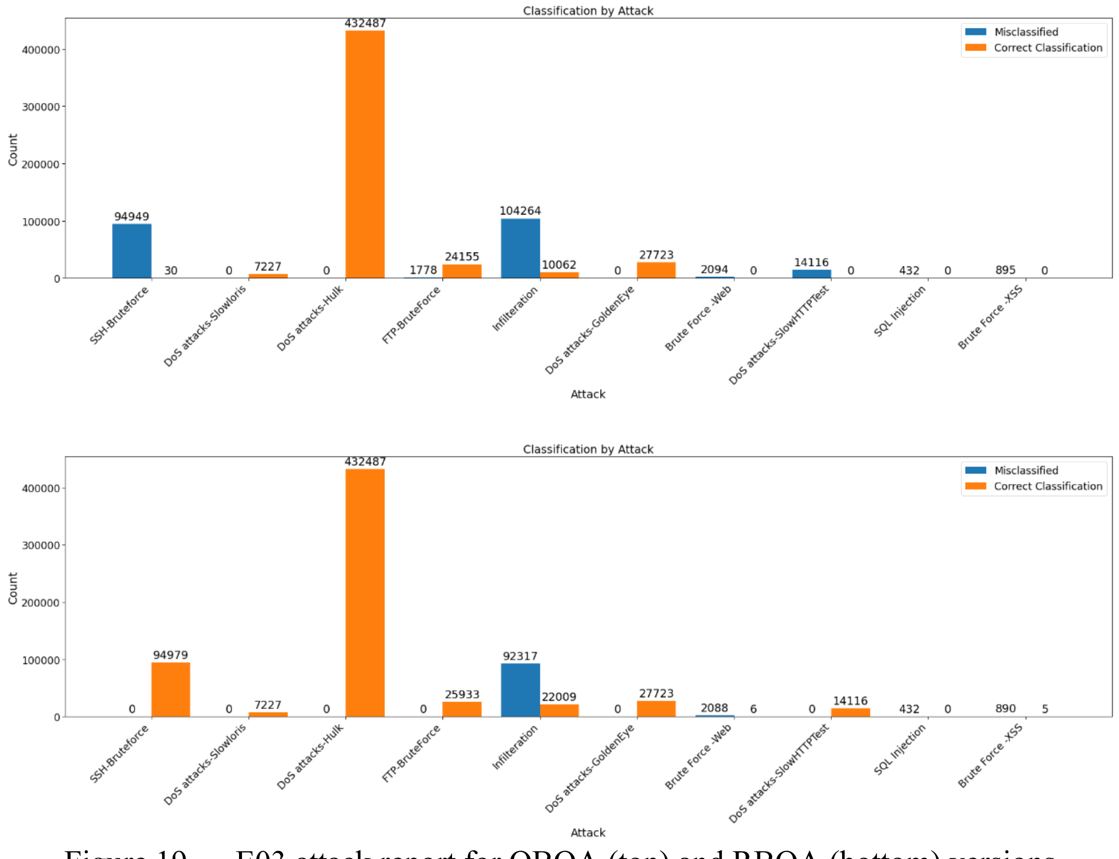


Figure 19 — E03 attack report for OBOA (top) and RBOA (bottom) versions

Since E03 is the best performing IDS for both OBOA and RBOA scenarios, its improvement can be observed. The OBOA version is only able to fully detect DoS/DDoS attacks using Slowloris, Hulk and GoldenEye. It can detect most brute force FTP attacks, and a miniscule number of brute force SSH and probe attacks. The RBOA version can detect all types of attacks except SQL injection. It can fully detect brute force via SSH, FTP attacks, and all DoS/DDoS attacks. The detection of probe attacks is double the OBOA version and only a few brute force web and XSS attacks are detected.

The confusion matrices between RBOA, OBOA versions of E01, E03 and E21 with default thresholds on the AR evaluation set is shown below. The performance boost between the OBOA and RBOA models is apparent. RBOA E01 performs well in the manner that is has the lowest number of FPs between any of the tested models. However, it also detects the lowest number of attacks. Between E03 and E21, E03 is the preferred IDS as it detects more attacks and has less FPs in both scenarios.

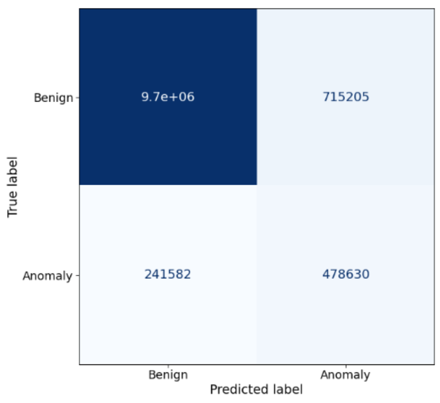
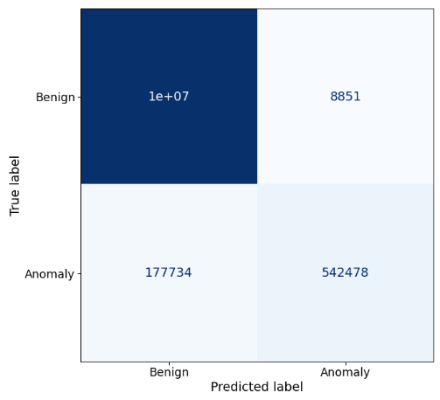
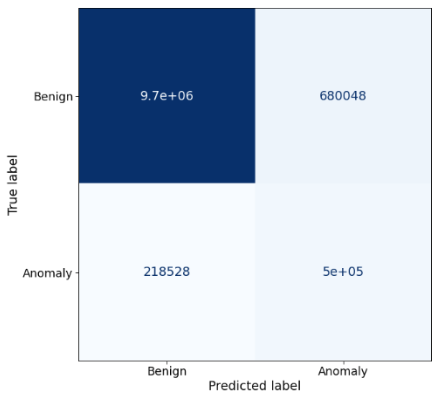
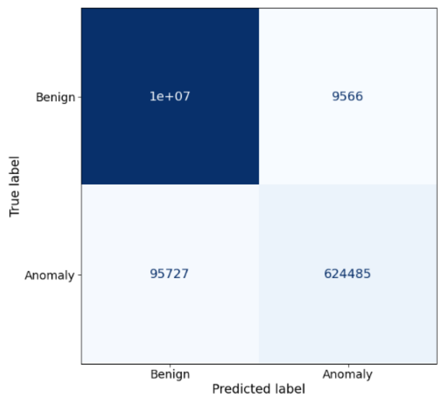
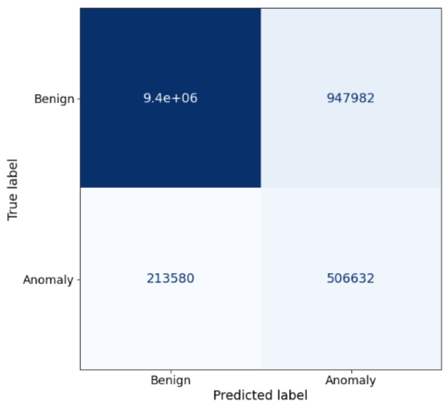
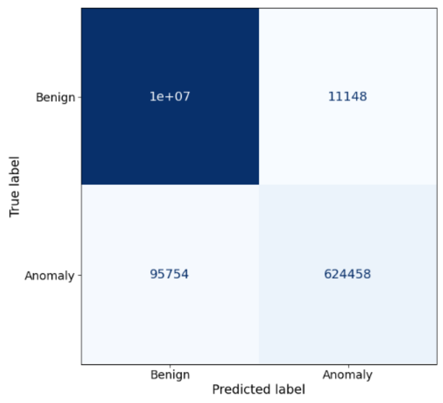
Model	**OBOA**	**RBOA**
E01		
E03		
E21		

Table 29 — Confusion Matrices for OBOA, RBOA E01, E03, E21 RF models

		OBOA		RBOA	
Model	**Metric**	**Test Set**	**Evaluation Set (AR)**	**Test Set**	**Evaluation Set (AR)**
E01	AUC	0.99	0.86	1	0.95
	F1 Score	0.98	0.48	0.99	0.85
	Precision	0.99	0.38	0.99	0.98
	Recall	0.98	0.67	0.99	0.75
	Specificity	0.99	0.93	0.99	0.99
E02	AUC	0.99	0.85	1	0.95
	F1 Score	0.98	0.50	0.99	0.85
	Precision	0.99	0.40	0.99	0.98
	Recall	0.97	0.69	0.99	0.76
	Specificity	0.99	0.93	0.99	0.99
E03	AUC	0.99	0.85	1	0.95
	F1 Score	0.98	0.52	0.99	0.92
	Precision	0.99	0.42	0.99	0.98
	Recall	0.97	0.70	0.99	0.87
	Specificity	0.99	0.93	0.99	0.99
E11	AUC	1	0.85	1	0.95
	F1 Score	0.98	0.52	0.99	0.85
	Precision	0.99	0.41	0.99	0.98
	Recall	0.97	0.69	0.99	0.76
	Specificity	0.99	0.93	0.99	0.99
E21	AUC	0.99	0.84	1	0.95
	F1 Score	0.98	0.47	0.99	0.92
	Precision	0.99	0.36	0.99	0.98
	Recall	0.97	0.70	0.99	0.86
	Specificity	0.99	0.91	0.99	0.99

Table 30 — Summary of OBOA and RBOA results for RF models

4.3.3. **Experiment 2 – IDS for Maximum Attack Coverage**

This experiment evaluates the developed RF models on the full evaluation set containing attacks unknown to the IDS. Since experiment 1 displays the superiority of the RBOA trained models, only the RBOA models are used in this experiment. It is important to note that because the features OUT_BYTES, OUT_DURATION, IN_PKTS, OUT_PKTS are included in data, the RF models discussed cannot be fairly compared to the DNN version of this experiment.

4.3.3.1. **Models**

The models used in this experiment are E03 and E01 from experiment 1. Refer to Table 28 for the parameters used when training these models. E03 is chosen for presenting the best results using the default threshold in experiment 1. E01 is chosen for having the highest AUC on the full evaluation set.

4.3.3.2. **Results**

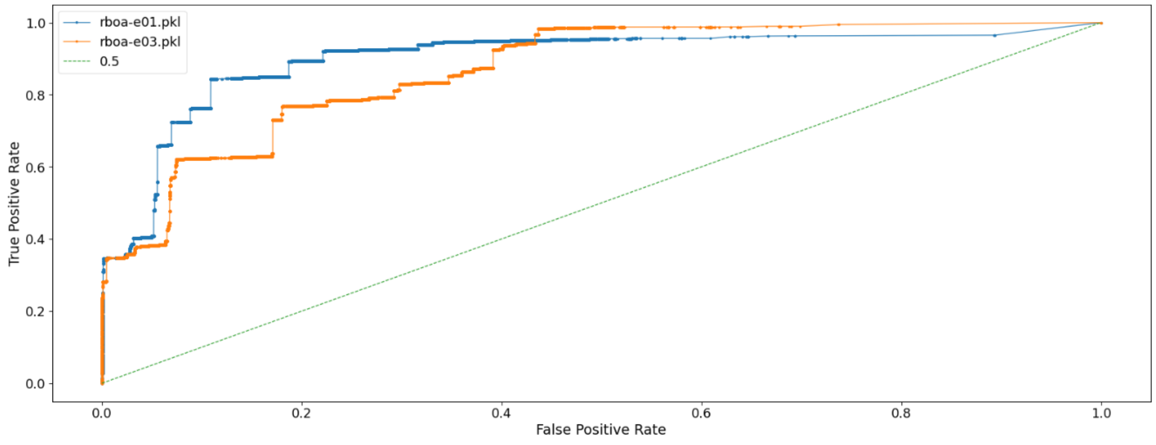


Figure 20 — ROC Curves of selected RF models on full evaluation set

Each model is evaluated two times on the full evaluation set. Once using the default classification threshold of 0.5 and another using the best classification threshold determined by Youdens J statistic on the ROC curve. Using the default threshold, neither model can detect LOIC, HOIC and botnet attacks. Similar results can be seen with DNNs. The difference is the DNNs can detect LOIC attacks, but this is purely because the features are manipulated to allow the model to do so. When the optimal threshold is determined for the RF, it is still unable to detect LOIC attacks.

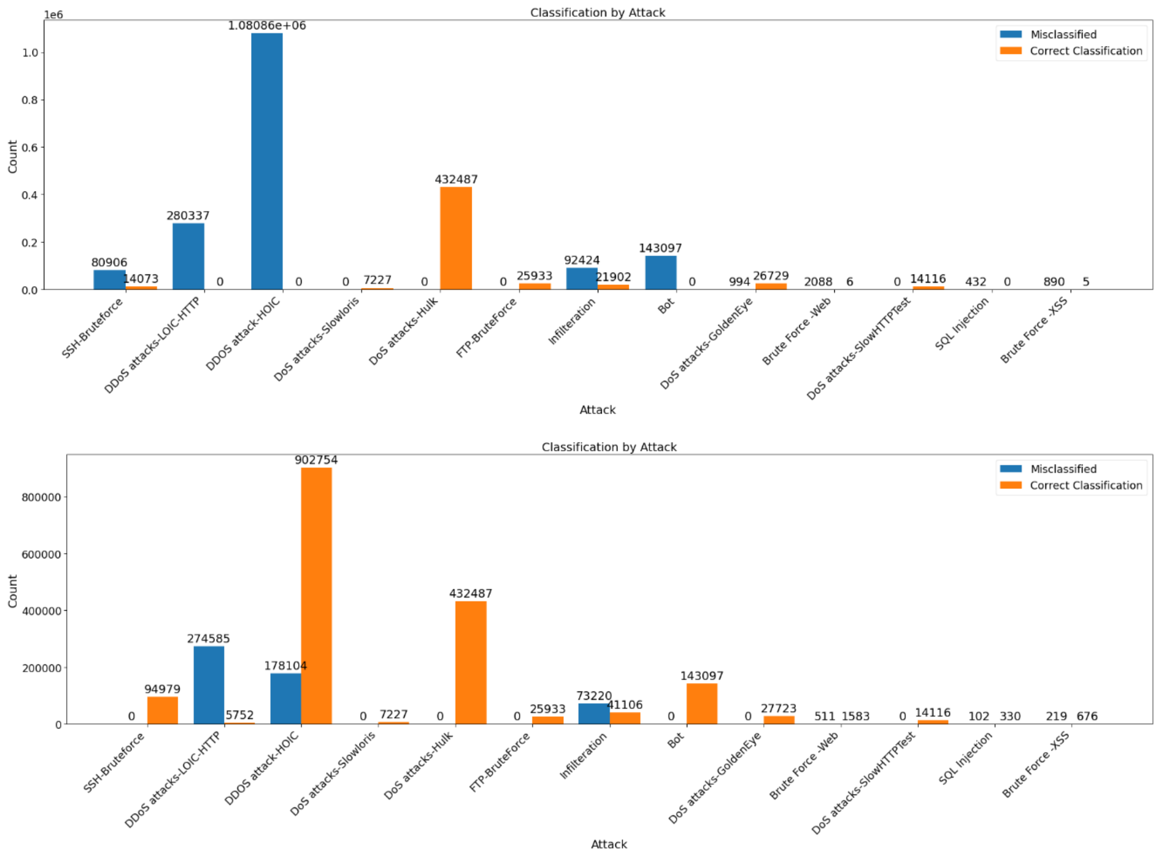


Figure 21 — Attack Report for E01 using threshold=0.5 (top) and optimal threshold (bottom)

Onwards, the adjusted threshold allows the models to detect most HOIC attacks and all botnet attacks. Most other attacks, such as XSS, SQL Injection, brute force web attacks are also detected. Probe attacks see a significant detection boost as well, but the missed probe attacks still outweigh the detected probe attacks. A common drawback between all threshold adjusted models is the increase in FPs.

An important factor to consider is also the optimal threshold. The optimal thresholds are extremely low (> 0.006). Since RF is an ensemble method, and uses voting to determine the output, such a low threshold translates to the output being reliant on 0.6% of the decision trees in the RF. For E03, using the optimal threshold means the output is reliant on 2 decision trees (400 × 0.005). For E01, the output is reliant on a single decision tree (300 × 0.003). Reliance on such few decision trees defeats the advantages provided by RF. Therefore, utilization of RF to maximize attack coverage using threshold tuning is not ideal. However, RF is a good option for detecting specific attack scopes without a custom threshold, as seen in the AR evaluation set. A summary of results is shown in Table 31. The adjusted threshold results are in grey.

Model	Threshold	Metric	Full Evaluation Set
E03	-	AUC	0.87
	0.5	F1 Score	0.43
		Precision	0.98
		Recall	0.28
		Specificity	0.99
	0.005369	F1 Score	0.57
		Precision	0.47
		Recall	0.75
		Specificity	0.81
E01	-	AUC	0.90
	0.5	F1 Score	0.39
		Precision	0.98
		Recall	0.24
		Specificity	0.99
	0.003404	F1 Score	0.67
		Precision	0.60
		Recall	0.76
		Specificity	0.89

Table 31 — RF Results for maximum attack coverage

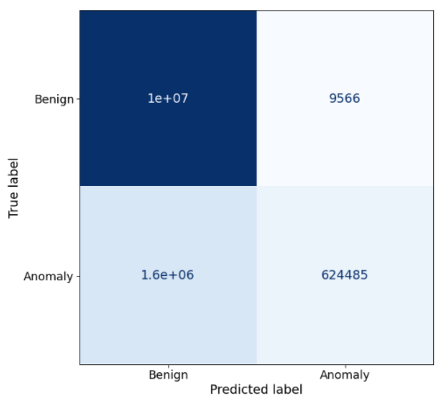
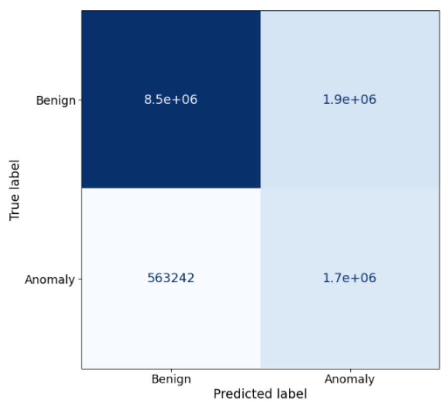
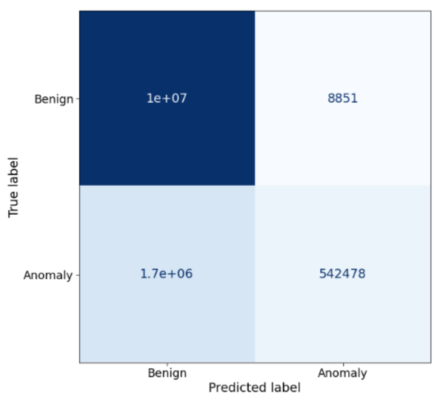
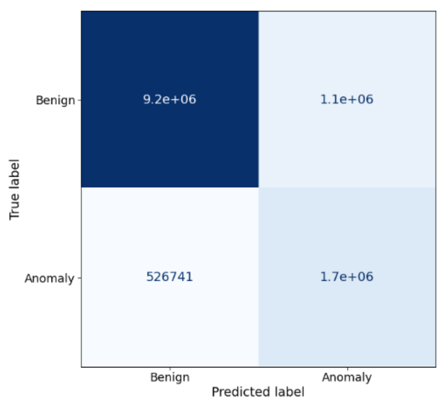
Model	**Threshold = 0.5**	**Threshold = Optimal Threshold**
E03 **Optimal Threshold = 0.005369**		
E01 **Optimal Threshold = 0.003404**		

Table 32 — Confusion Matrices for RF Maximum Attack Detection

Chapter 5 — **Conclusion & Future Work**

5.1. **Conclusion**

In this thesis, the transferability of NetFlow data from a foreign network for a machine learning based IDS is explored. This research demonstrates that in scenarios where absolutely no data is available from the target network for IDS training, data from a foreign source or global repository can be used instead. If benign data from the target network is available, but attack data is not, then the benign data from the target network can be mixed with attack data from a foreign network to create a powerful IDS. In addition, this thesis studies situations where the IDS is faced with detecting unique attacks it has never seen before, and methods to boost its performance for such scenarios.

Four IDS datasets are used, each which represents a unique purpose network and contains unique attacks. Three of the four datasets are used to train the IDS, and the last dataset is used to evaluate the IDS. This simulates a scenario where an IDS trained completely on foreign network data is applied onto a separate, independent network. It is studied whether this IDS training scheme can provide adequate attack coverage for a certain scope of attacks. The IDSs developed in this thesis are trained on mostly IoT network traffic, and the evaluation network is a conventional wired network.

The first experiment concludes the transferability of both benign and anomalous traffic from foreign networks (OBOA training data). IDSs trained on benign, anomalous data from completely different networks can roughly distinguish benign and anomalous target network traffic. However, such IDSs output a large number of FPs and have a limited scope of attack detection. Another training scheme is tested to address these issues where the benign data used to train the IDS originates from the network the IDS is to be implemented on, and the attack data is from foreign networks (RBOA training data). This method not only significantly decreases the number of FPs, but broadens the IDSs scope of attacks, allowing for more attacks to be detected. In addition, this scheme allows most machine learning models to succeed as IDSs, whereas the other training scheme requires strict machine learning hyperparameter tuning. The closing recommendations from this experiment is training an IDS on target network benign data in conjunction with foreign network attacks provides the best results for scenarios where collecting target network attack data is logistically difficult. If collecting benign data is also an issue, using foreign network benign traffic for IDS training is a worst case scenario option, and the IDS will be far less reliable.

The second experiment evaluates IDS detection on attack types/tools which have zero similarity and representation in the IDS training data. This experiment aims to improve detection for zero-day attacks and attacks the IDS is unfamiliar with. By using RBOA training data, feature selection and ROC analysis, the IDSs successfully have an increased detection scope.

Both experiments are completed using DNNs and RF. Comparing both models on experiment 1, DNN is preferred for when using OBOA training data, but the models hyperparameters must be perfectly tuned. When using RBOA training data, RF is the preferred scheme as the average model easily achieves a higher F1 score compared to DNN with more lenient tuning. Experiment 2, development of IDS attack coverage, demonstrates RF is not an ideal machine learning algorithm with threshold tuning because the RF becomes reliant on one or two decisions trees. Thus, DNNs are the preferred algorithm while using threshold tuning to detect new attacks.

5.2. **Future Work**

The work from this thesis has unlocked a variety of future research ideas for IDS development.

The first area of research is to determine an optimal method to distribute attack data in the IDS training set. This thesis used an algorithm to automatically distribute the attack data, based on the distribution in the original training datasets. This led to unbalanced distributions of attack types in the training data. For example, 45% of the total attack data is DoS/DDoS, while probes represent 25%, and brute force attacks represent 30%. Ultimately, the task is to identify a method to distribute these attacks and tools in the training data to maximize IDS performance.

The second area of research is determining the similarity of different attack tools. In this thesis, the IDS training set did not contain any HULK, Slowloris, or Slowhttptest DoS/DDoS data, yet the developed IDSs easily detected these tools. Other tools of similar nature like HOIC, LOIC and botnet were not detected by the IDS unless using feature selection or threshold tuning. It can be concluded that some attack tools operate like other tools, while others may not operate in a similar manner. Identification of similar tools can streamline IDS training data generation and reduce redundancy within it.

The third area of research is dataset shift for IDS. The training set for the DNN IDS in this thesis is composed of 80% benign data. The distribution of data in production is ever changing and varies from the distribution in the training data. This has an impact on the IDS performance. This is a foreseeable issue as this will cause false IDS predictions and result in unreliability.

The final recommendation is to improve existing IDS datasets. The UQ-NIDS-v2 dataset is a step in right direction by unifying multiple IDS datasets under the same features. However, there is work available to improve this dataset. For instance, labelling which tool is used for which attacks is a tremendous benefit. This would allow for easier analysis of the data, see which attacks are similar, and provide more control for training data generation and filtering.

BIBLIOGRAPHY

[1] A. A. Ghorbani, W. Lu and M. Tavallaee, Network Intrusion Detection, Fredericton: Springer Science, 2010.

[2] J. Quittek, T. Zseby, B. Claise and S. Zander, "Requirements for IP Flow Information Export (IPFIX) RFC 3917," IETF, [Online]. Available: https://tools.ietf.org/html/rfc3917.

[3] A. Sperotto, G. Schaffrath, R. Sadre, C. Morariu, A. Pras and B. Stiller, "An Overview of IP Flow-Based Intrusion Detection," *IEEE COMMUNICATIONS SURVEYS & TUTORIALS,* vol. 12, no. 3, pp. 343-356, 2010.

[4] D. Xu and Y. Tian, "A Comprehensive Survey of Clustering Algorithms," *Annals of Data Science,* vol. 2, pp. 165-193, 2015.

[5] L. v. d. Maaten, E. Postma and J. v. d. Herik, "TiCC TR 2009–005: Dimensionality Reduction: A Comparative," Tilburg University, Tilburg, The Netherlands, 2009.

[6] R. T. T. Hastie and J. Friedman, in *The Elements of Statistical Learning Second Edition*, Springer, 2009, p. 588.

[7] H. HINDY, D. BROSSET, E. BAYNE, A. SEEAM, C. TACHTATZIS, R. ATKINSON and X. BELLEKENS, "A Taxonomy of Network Threats and the Effect of Current Datasets on Intrusion Detection Systems," *IEEE Access,* vol. 8, no. 104650–104675, 2020.

[8] S. Tsimenidis, T. Lagkas and K. Rantos, "Deep Learning in IoT Intrusion Detection," *Journal of Network and Systems Management,* vol. 20, no. 00408, 2020.

[9] N. C. N. P. W. e. a. Sultana, "Survey on SDN based network intrusion detection system using machine learning approaches.," *Peer-to-Peer Networking and Applications ,* vol. 12, p. 493–501, 2019.

[10] L. M. S. M. H. J. Mohamed Amine Ferrag, "Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study," *Journal of Information Security and Applications,* vol. 50, no. 102419, 2020.

[11] H. Liu and B. Lang, "Machine Learning and Deep Learning Methods for Intrusion Detection Systems: A Survey.," *Appl. Sci,* vol. 9, no. 4396, 2019.

[12] X. B. A. H. C. T. a. R. A. E. Hodo, "Shallow and Deep Networks Intrusion Detection System: A Taxonomy and Survey," *arXiv:1701.02145.,* pp. 1-43, 2017.

[13] N. Farnaaz and M.A.Jabbar, "Random Forest Modeling for Network Intrusion Detection System," *Procedia Computer Science,* vol. 89, pp. 213-217, 2016.

[14] M. A. M. Hasan, M. Nasser, B. Pal and S. Ahmad, "Support Vector Machine and Random Forest Modeling for Intrusion Detection System (IDS)," *Journal of Intelligent Learning Systems and Applications,* vol. 6, pp. 45-52, 2014.

[15] E. Min, J. Long, Q. Liu, J. Cui and Chen, "TR-IDS: Anomaly-Based Intrusion Detection through Text-Convolutional Neural Network and Random Forest," *Secur. Commun. Netw.,* vol. 2018, 2018.

[16] R. VINAYAKUMAR, M. ALAZAB, K. P. SOMAN, P. POORNACHANDRAN, A. A.-N. and S. VENKATRAMAN, "Deep Learning Approach for Intelligent Intrusion," *IEEE Access,* vol. 7, pp. 41525 - 41550, 2019.

[17] F. Y. Yavuz, D. Ünal and E. Gül, "Deep Learning for Detection of Routing Attacks in the Internet of Things," *International Journal of Computational Intelligence Systems,* vol. 12, pp. 39-58, November 2018.

[18] S. Potlur, S. Ahmed and C. Diedrich, "Convolutional Neural Networks for Multi-class intrusion detection system," in *International Conference on Mining Intelligence and Knowledge Exploration*, 2018.

[19] A. I. N. M. C. a. Y. E. M. Tan, "A Neural Attention Model for Real-Time Network," in *IEEE 44th Conference on Local Computer Networks (LCN)*, Osnabrueck, Germany, 2019.

[20] R.-H. Hwang, M.-C. Peng, V.-L. Nguyen and Y.-L. Chang, "An LSTM-Based Deep Learning Approach for Classifying Malicious Traffic at the Packet Level," *Appl. Sci.,* vol. 9, no. 3414, 2019.

[21] C. YIN, Y. ZHU, J. FEI and X. HE, "A Deep Learning Approach for Intrusion Detection Using Recurrent Neural Networks," *IEEE Access,* vol. 5, pp. 21954-21961, 2017.

[22] R.-H. HWANG, M.-C. PENG, C.-W. HUANG, P.-C. LIN and V.-L. NGUYEN, "An Unsupervised Deep Learning Model for Early Network Traffic Anomaly Detection," *IEEE Access,* vol. 8, no. 30399, pp. 30387-, 2020.

[23] Y. ZHANG, P. LI and X. WANG, "Intrusion Detection for IoT Based on Improved Genetic Algorithm and Deep Belief Network," *IEEE Access,* vol. 7, pp. 31711-31722, 2019.

[24] Q. Tian, D. Han, K.-C. Li, X. Liu, L. Duan and A. Castiglione, "An intrusion detection approach based on improved deep belief network," *Applied Intelligence,* vol. 50, p. 3162–3178, 2020.

[25] G. C. Fernandez and S. Xu, "A Case Study on Using Deep Learning for Network Intrusion Detection," in *IEEE Military Communications Conference (MILCOM)*, Norfolk, VA, USA, 2019.

[26] M. AL-Hawawreh, N. Moustafa and E. Sitnikova, "Identification of malicious activities in industrial internet of things based on deep learning models," *Journal of Information Security and Applications,* vol. 41, pp. 1-11, 2018.

[27] B. Zhang, Y. Yu and J. Li, "Network Intrusion Detection Based on Stacked Sparse Autoencoder and Binary Tree Ensemble Method," in *IEEE International Conference on Communications Workshops (ICC Workshops)*, Kansas City, MO, USA, 2018.

[28] H. Zhang, C. Q. Wu, S. Gao, Z. Wang, Y. Xu and Y. Liu, "An Effective Deep Learning Based Scheme for Network Intrusion Detection," in *2018 24th International Conference on Pattern Recognition (ICPR)*, Beijing, China, 2018.

[29] H. Zhang, Y. Li, Z. Lv, A. K. Sangaiah and T. Huang, "A real-time and ubiquitous network attack detection based on deep belief network and support vector machine," *IEEE/CAA Journal of Automatica Sinica,* vol. 7, pp. 790-799, 2020.

[30] M. Sarhan, S. Layeghy, N. Moustafa and M. Portmann, "Towards a Standard Feature Set of NIDS Datasets," University of Queensland, Brisbane, 2021.

[31] N. M. a. J. Slay, "UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," 2015 Military Communications and Information Systems Conference (MilCIS), Canberra, ACT,, 2015.

[32] I. Sharafaldin, A. H. Lashkari and A. A. Ghorbani, "CSE-CIC-IDS2018 on AWS," University of New Brunswick, 2018. [Online]. Available: https://www.unb.ca/cic/datasets/ids-2018.html; https://registry.opendata.aws/cse-cic-ids2018/. [Accessed January 2021].

[33] A. Alsaedi, N. Moustafa, Z. Tari, A. Mahmood and A. Anwar, "TON_IoT Telemetry Dataset: A New Generation Dataset of IoT and IIoT for Data-Driven Intrusion Detection Systems," *IEEE Access,* vol. 8, pp. 165130-165150, 2020.

[34] N. Koroniotis, N. Moustafa, E. Sitnikova and B. Turnbull, " Towards the development of realistic botnet dataset in the Internet of Things for network forensic analytics: Bot-IoT dataset," *Future Generation Computer Systems,* vol. 100, pp. 779-796, 2019.

[35] "Hulk Flood," MazeBolt, [Online]. Available: https://kb.mazebolt.com/knowledgebase/hulk-flood/.

[36] "GoldenEye HTTP Flood," MazeBolt, [Online]. Available: https://kb.mazebolt.com/knowledgebase/goldeneye-http-flood/.

[37] "Slowloris DDoS attack," CloudFlare, [Online]. Available: https://www.cloudflare.com/en-ca/learning/ddos/ddos-attack-tools/slowloris/.

[38] "SlowHTTPTest Package Description," Kali.org, [Online]. Available: https://tools.kali.org/stress-testing/slowhttptest.

[39] "High Orbit Ion Cannon (HIOC)," imperva, [Online]. Available: https://www.imperva.com/learn/ddos/high-orbit-ion-cannon/.

[40] "Low Orbit Ion Cannon (LOIC)," imperva, [Online]. Available: https://www.imperva.com/learn/ddos/low-orbit-ion-cannon/.

[41] epsylon, "ufonet," Github, [Online]. Available: https://github.com/epsylon/ufonet.

[42] "hping3 Package Description," Kali.org, [Online]. Available: https://tools.kali.org/information-gathering/hping3.

[43] P. Paganini, "Researchers from WootCloud Labs have uncovered a new IoT botnet named Ares that is targetting Android-based devices.," 31 August 2019. [Online]. Available: https://securityaffairs.co/wordpress/90624/malware/ares-iot-botnet.html.

[44] "nmap.org," [Online]. Available: https://nmap.org/.

[45] "The Nessus Family," tenable, [Online]. Available: https://www.tenable.com/products/nessus. [Accessed 5 August 2021].

[46] O. Arkin and F. Yarochkin, "Xprobe v2.0 A "Fuzzy" Approach to Remote Active Operating System Fingerprinting," 2002.

[47] lanjelot, "patator," [Online]. Available: https://github.com/lanjelot/patator.

[48] Ettercap, "Ettercap," [Online]. Available: https://github.com/Ettercap/ettercap.

[49] digininja, "CeWL," [Online]. Available: https://github.com/digininja/CeWL.

[50] "Hydra Package Description," Kali.org, [Online]. Available: https://tools.kali.org/password-attacks/hydra.

[51] epsylon, "xsser," [Online]. Available: https://github.com/epsylon/xsser.

[52] "Quick Start Guide," Rapid7, [Online]. Available: https://docs.rapid7.com/metasploit/quick-start-guide.

[53] J. G. Moreno-Torres, T. Raeder, R. Alaiz-Rodríguez, N. V. Chawla and F. Herrera, "A unifying view on dataset shift in classification," *Pattern Recognition,* vol. 45, no. 1, pp. 521-530, 2012.

[54] P. Probst and A.-L. Boulesteix, "To tune or not to tune the number of trees in random forest?," arXiv 1705.05654, 2017.

[55] "Random Search for Hyper-Parameter Optimization," *Journal of Machine Learning Research,* vol. 13, no. 281-305, 2012.

APPENDICES

Appendix A: GITHUB

The results and notebook files are available on GitHub at the following URL: https://github.com/wmati/Transferability-of-Netflow-Data-for-IDS. This repository does not include the training, validation, testing and evaluation dataset files.

VITA AUCTORIS

NAME:	William Mati
PLACE OF BIRTH:	Windsor, ON, Canada
YEAR OF BIRTH:	1996
EDUCATION:	Holy Names High School, Windsor, ON, 2014
	University of Windsor, B.A.Sc., Windsor, ON, 2019
	University of Windsor, M.A.Sc., Windsor, ON, 2021