An Entropy-based approach to Network Attack Classification with Deep Neural Network

by

Emily H. Do
S.B., M.I.T., 2016

Submitted to the
Department of Electrical Engineering and Computer Science
in Partial Fulfillment of the Requirements for the Degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

Massachusetts Institute of Technology

September 2019

Author: ____
Department of Electrical Engineering and Computer Science
August 23, 2019

Certified by: ____
Dr. Vijay N. Gadepally
Senior Member of the Technical Staff
Thesis Supervisor
August 23, 2019

Accepted by: ____
Katrina LaCurts
Chair, Master of Engineering Thesis Committee

An Entropy-based approach to Network Attack Classification with Deep Neural Network

by

Emily H. Do

Submitted to the Department of Electrical Engineering and Computer Science
on August 23, 2019
in partial fulfillment of the requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

Detecting and classifying attacks on computer networks is a significant challenge for network providers and users. This thesis project builds a deep neural network to detect and classify network attacks. Our approach based on the hypothesis that each type of network attacks generates a distinguishable change in the entropies of certain features of network flows. To generate a training and validation dataset, synthetic attacks of different types and levels of intensity are injected to the MAWI dataset, which contains captured raw network traffic from an Internet backbone link. Experimental results show that our machine learning model can achieve a high accuracy for network attacks which intensity is as low as 6% of the original traffic. This result is very promising, considering the fact that the amount of traffic in the Internet backbone link is substantial. This work also evaluates and quantifies the model performances on different attack types and the levels of intensity of these attacks.

Thesis Supervisor: Dr. Vijay N. Gadepally
Title: Senior Member of the Technical Staff, MIT Lincoln Laboratory

Acknowledgements

Many thanks go to the following people and organizations (in no particular order).

Vijay Gadepally for your guidance and support. I have learned so much from you, and this thesis would not be completed without your technical and content review.

Jeremy Kepner for showing me the ways to make this thesis happened.

Lauren Milechin and Adam Michaleas for assisting me with MIT SuperCloud.

Carlos Garcia Cordero and Jens Keim for all of your informative and prompt correspondences in regards of ID2T.

Anne Hunter for making the MEng administration process a pleasant experience.

Raytheon BBN Technologies for providing me the opportunity to further my education.

My managers and colleagues for your encouragement and inspiration.

MIT for believing in me and giving me the opportunities to pursue my academic dreams.

Nancy, Linh, John, Natalie and Lam Nguyen for all of your support during my first years in America. You are the best aunts and uncles.

My mom for always being there for me.

Contents

Acknowledgements **5**

Contents **7**

List of Figures **9**

List of Tables **11**

1. Introduction **13**

2. Related Work **17**

2.1. Deep Learning Models for anomaly detection 17

2.2. Unsupervised Learning for Anomaly Detection 18

2.3. MAWILab Dataset 23

3. Methods **25**

3.1. Data processing pipeline 25

3.2. Dataset 26

3.3. Synthetic attack injection 27

3.4. Network flow extractor 29

3.5. Feature extraction 31

4. Synthetic Attack injections **33**

4.1. Distributed Denial of Service (DDoS) Attack 33

4.2. Port Scan Attack 34

4.3. Point-to-Point Denial of Service (DoS) Attack 35

4.4. Network Scan 36

5. Machine Learning model 39

5.1. Training dataset and Validation Dataset 39

5.2. Input Features 39

5.3. Classification Model 40

6. Evaluation 43

7. Computational Performance 55

7.1. Synthetic attack injection time 55

7.2. Pre-processing time 56

7.3. Training time and prediction time 58

8. Conclusions and Future work 59

Bibliography 61

List of Figures

Figure 1. Replicator Neural Network 19

Figure 2. Anomaly Score Changes due to DDoS attacks 20

Figure 3. Anomaly Score Changes due to Port scan attacks 21

Figure 4. Data processing pipeline 26

Figure 5. Anomaly Scores for the original traffic and traffic with injected attacks 28

Figure 6. YAFSCII file in tabular format 30

Figure 7. The Deep Neural Network for attack detection and classification 41

Figure 8. Confusion matrix for validation data 43

Figure 9. Confusion matrix for validation data injected with high and moderate intensity attacks 45

Figure 10. Mixed confusion matrix for different types of attack and intensity 46

Figure 11. Attack intensity vs. Accuracy for DDoS attacks 47

Figure 12. Attack intensity vs. Accuracy for Port scan attacks 48

Figure 13. Attack intensity vs. Accuracy for point-to-point DoS attacks 49

Figure 14. Attack intensity vs. Accuracy for Network scan attacks 50

Figure 15. Percentage of injected traffic vs. Accuracy 52

Figure 16. Synthetic attack injection time with ID2T 56

Figure 17. Pre-processing time 57

List of Tables

Table 1. Features of interest from network flows 32

Table 2. ID2T parameters for DDoS attacks at different intensities... 34

Table 3. ID2T parameters for Port scan attacks at different intensities....................................... 35

Table 4. ID2T parameters for point-to-point DoS attacks at different intensities 36

Table 5. ID2T parameters for SMB Network scan attacks at different intensities 37

1. Introduction

Detecting and classifying attacks on computer networks is a significant challenge for network providers and users. Network traffic volumes continue to increase rapidly while network attacks become more prevalent and sophisticated. Current efforts for attack detection and classification largely rely on heuristics or rule-based methods that do not scale with changes in network traffic patterns or network attack patterns. Developing efficient methods and tools for detecting and classifying network attacks becomes necessary.

In our conversations with a number of network engineers and cyber-security analysts from various organizations, it is clear that there is a large demand for automated tools that can measure and predict anomalous behavior in computer networks. At the same time, analysts are weary of tools that have high rates of false alarms or do not provide sufficient feedback on the type of attack or more information that can be used to address the issue at hand.

Over the past two decades, Deep Learning [1] has become a major force in revolutionizing and driving forward Machine Learning and Artificial Intelligence. Deep learning benefits greatly from the exponential growth of computing power, the availability of big training datasets and improvements in learning algorithms. Deep Learning methods have played a major role in technology developments in many different fields such as image recognition, speech recognition, natural language understanding, medical diagnosis, and self-driving cars [1]. In recent years, they have also developed and applied to cybersecurity, in which network anomaly detection is a fundamental topic. As defined by Chandola et al., "anomalies are patterns in data that do not conform to a well-defined notion of normal behavior" [2]. Anomalies in network traffic can be indicators of malicious activities or errors. Learning to detect anomalies in network traffic is challenging due to the large amount of raw traffic data which is often either insufficiently labelled or unlabeled [2].

There has been research performed to explore different deep learning models, such as Fully Connected Network, Variational Autoencoder, Long Short-Term Memory with Sequence to Sequence [3], Convolution Neural Networks [4] [5] [6] [7] and Replicator Neural Networks [8] [9] [10] [11], to network anomaly detection.

This thesis builds a deep learning model to detect and classify network attacks from raw network traffic. Our approach based on the hypothesis that each type of network attacks generates a distinguishable change in the entropies of certain features of network flows. This thesis explores the effectiveness of network flow entropies, a property that is commonly utilized in anomaly detection [9] [10] [11] [12], to detect and classify network attacks. This work also evaluates and quantifies the model performances on different attack types and the intensity levels of these attacks.

Given the difficulty in finding network traffic data with clear labelled anomalies, we leverage a synthetic data generator [13] [14] to inject synthetic attacks into real network traffic. The use of synthetic attacks allows us to create a labeled dataset containing simulated attacks of desire types and with desire intensities. In this experiment, we inject synthetic attacks of different types and with different levels of intensity into an Internet backbone link in the MAWI dataset [15] and use the dataset to train a deep neural network to detect and classify network attacks.

This work, however, is restricted by the current limited capabilities of the synthetic attack injection tool. In addition, the fact that the MAWI dataset does not contain any application payload [15] excludes our model from classifying the attacks that can only be identified and/or distinguished by their packet payloads. Therefore, this thesis focuses on detecting and classifying 4 types of attacks: DDoS, port scan, point-to-point DoS, and network scan.

2. Related Work

2.1. Deep Learning Models for anomaly detection

There has been research performed to explore different deep learning models. For instances, Malaiya et al. [3] evaluated 3 different deep learning models on network anomaly detection: Fully Connected Network, Variational Autoencoder, and Long Short-Term Memory with Sequence to Sequence (LSTM-Seq2Seq). Their results showed that models based on the LSTM-Seq2Seq structure can achieve high accuracy [3]. Kwon et al. performance [4] evaluated Convolution Neural Networks (CNNs) and observed that the CNN models perform better than the Variational Autoencoder models in some instances, but do not perform better than the Fully Connected Network models or the LSTM-Seq2Seq models. In addition, they evaluated three CNN models with different internal depths to determine their impacts on performance. Their results suggested that simply adding more layers to network models does not improve the performance [4]. Several other research also explored CNN-based models for network intrusion [5] [6] [7].

In a separate effort, Cordero et al. proposed a method to detect scanning attacks using Replicator Neural Networks, which are particular instances of auto-encoders. Their method used the entropies of 4 features of network flows as the input to the Replicator Neural Network [9]. To follow up this work, Kotani et al. proposed a method called robust auto-encoders which split the training data into benign and outlier features repeatedly in the learning process. Their experiments showed lower false positive rates [10]. To improve the auto-encoder models, Li et al. proposed using the Bayesian probability model to predict the parameters for the auto-encoders. Their results indicate that the new Stacked Auto-Encoder improves the model accuracy in comparison to the original Stacked Auto-Encoder [11].

Recently, there is also research on the generalization of neural network models on structured datasets. For instance, Graph Convolutional Networks (GCNs) are deep network models applied to graphs [16] [17] [18]. Fast Learning with Graph Convolutional Networks (Fast-GCN) is a promising deep learning method applied to graphs [19]. Fast-GCN has showed to have shorten training times and better accuracy [17] [19] and it has been applied to multiple domains [20].

2.2. Unsupervised Learning for Anomaly Detection

In *Analyzing Flow-based Anomaly Intrusion Detection using Replicator Neural Networks* by Cordero et al. [9], the authors describe a methodology using Replicator Neural Network (RNN) to perform anomaly detection in network traffic. Their method is shown to be robust to certain types of attack and has inspired parts of this work. In their RNN model, the Shannon entropies (H) of 4 features (source IP, source port, destination IP, destination port) of the network flows are fed into a Replicator Neural Network as depicted in Figure 1. This unsupervised model learns to reproduce the input features.

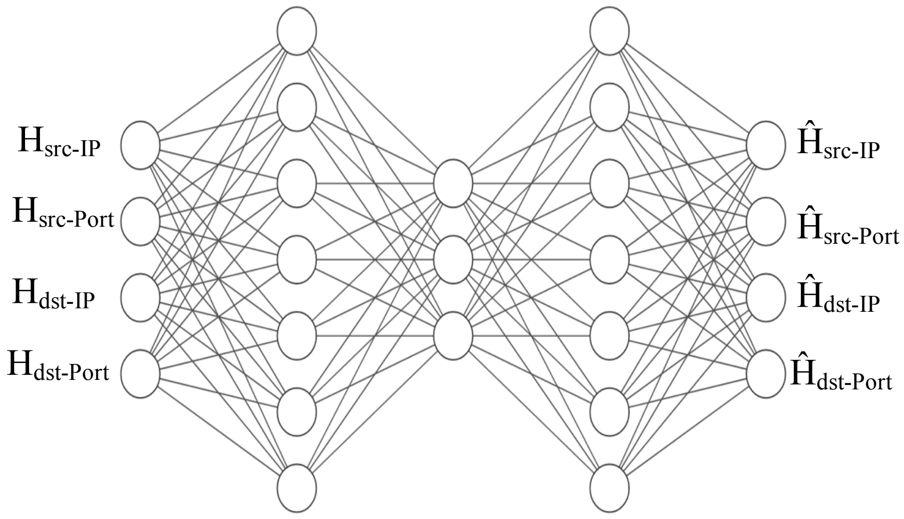


Figure 1. Replicator Neural Network: The network model used by Cordero et al. to detect network anomaly. It consists of an input layer (4 nodes), 3 hidden layers (7 nodes, 3 nodes, and 7 nodes) and an output layer (4 nodes). The input features to the model are the entropies of source IP, source port, destination IP, and destination port. The network learns to reproduce the same input features.

Anomaly Scores (AS) are defined to be the mean Euclidean distance between the input and the prediction. In this equation, D is the number of features and $x = (x_1,\ x_2, \dots ,\ x_D)$ is an input feature vector.

$$AS(x) = \frac{1}{D}\sum_{i=1}^{D}(x_i - \widehat{x_i})^2$$

If an input to the RNN is anomalous, the network outputs a prediction that is very different from the input and its Anomaly Score is high. A threshold is determined in the training phase, an input is considered to be anomalous if its Anomaly Score is greater than the pre-determined threshold. Otherwise, the input is considered normal.

To test the model, two types of synthetic attacks (DDoS and Port Scan) with 3 different levels of intensity were injected into the original network traffic on one specific time window. Figure 2 and Figure 3 show the Anomaly Score changes due to DDoS and Port Scan attacks, respectively.

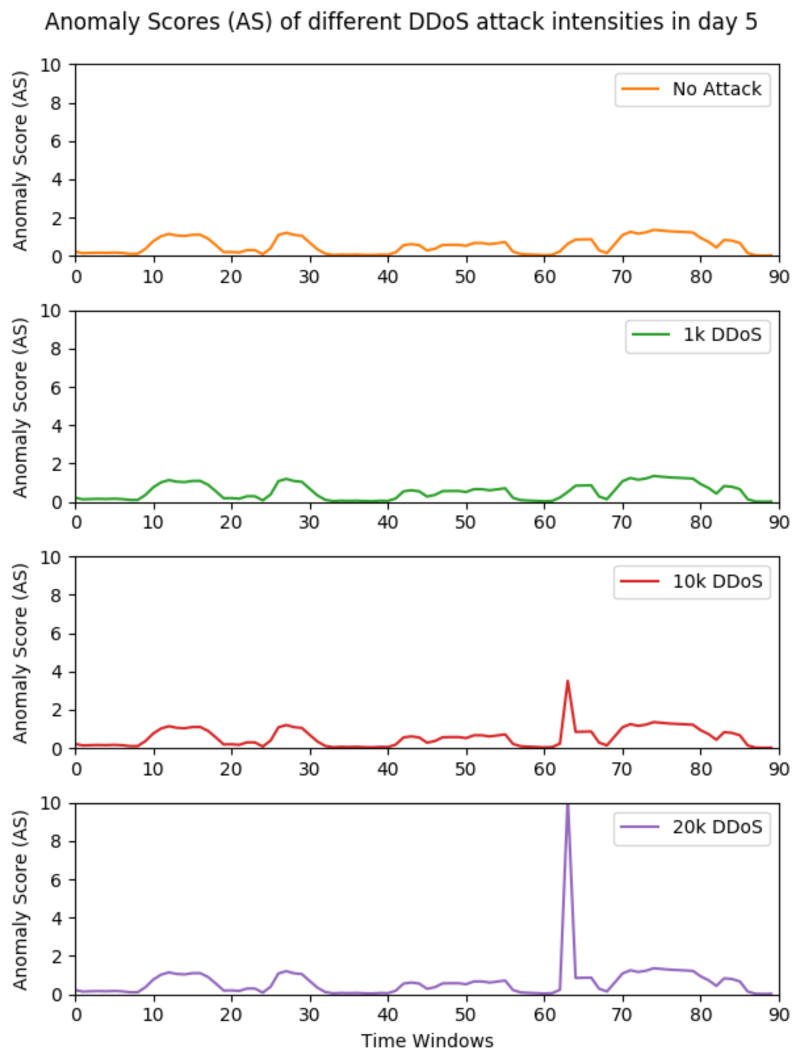


Figure 2. Anomaly Score Changes due to DDoS attacks: These plots show the Anomaly Score changes when DDoS attacks of 3 different intensities (1k, 10k and 20k packets per second) are injected into the same time window on the day 2015-04-03 of the MAWI dataset.

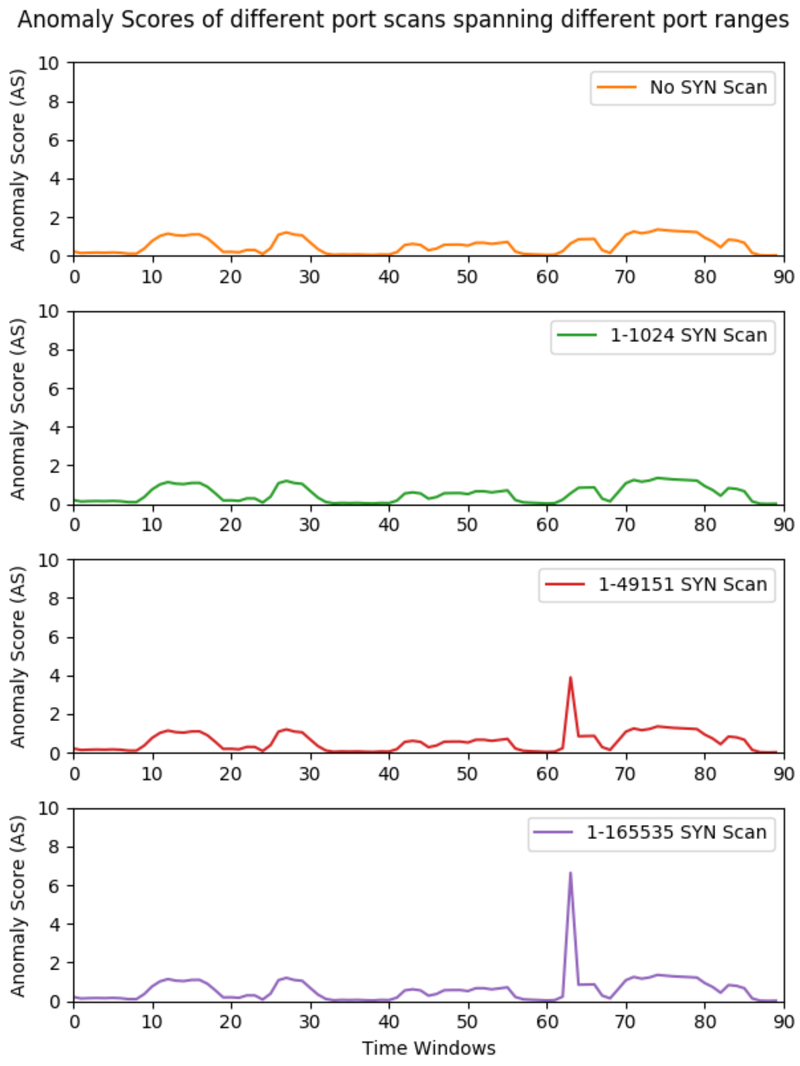


Figure 3. Anomaly Score Changes due to Port scan attacks: These plots show the Anomaly Score changes when SYN port scan attacks of 3 different intensities (scanning 1024 ports, 49151 ports and 65535 ports in 5 seconds) are injected into the same time window on the day 2015-04-03 of the MAWI dataset.

The anomaly detection method using Replicator Neural Network cannot classify the types of the attacks. It can predict something is anomalous, but it does not know which type of attack is occurring. The method required a threshold to determine if an attack is anomalous, thus its performance accuracy depends on how the threshold is chosen. Nevertheless, no experiment was performed to quantify the model accuracies on different thresholds, different attacks and different intensities. The experiment only used data from 5 days with 2 types of attacks injected to a single time window.

While this technique is useful in detecting anomalous behavior in network traffic, it does not classify what types of attack may be occurring nor provide any additional feedback that could be used by a network analyst. In addition, the existing techniques are largely dependent on the selection of an arbitrary threshold value to indicate an attack. If the threshold is too low, the false alarm rate is high and if the threshold is too high, the probability of detection is low. Furthermore, the range of anomaly scores vary from attack to attack and network to network. In this thesis, we extend existing research into anomaly detection in the following ways:

(i) The capability of classifying the types of the network attacks: The output of our system goes beyond a score to classify the type of attack;

(ii) Threshold independency: Our system does not require selection of a threshold;

(iii) Evaluation of the model performances on different types of attack and level of intensity: We have done an exhaustive analysis of various attacks and intensity levels to characterize classifier performance.

2.3. MAWILab Dataset

In the quest of finding a good labeled dataset for network attack classification, we first attempted to use the MAWILab [21] [22]. The MAWILab contains a database of labels of anomalous traffic in the MAWI dataset that are obtained using a graph-based methodology that combines results from different anomaly detectors [21][22]. It is publicly available and is intended to assist cybersecurity researchers by providing them with labeled datasets.

Because our classification method segments network flows into time windows, it is imperative that the labels contain the exact time of the anomaly. The MAWILab labels contain timestamps of the start and the end of the anomaly. However, for a large portion of the labeled anomalies, these timestamps are invalid, in which the start timestamp is set to 0, and the end timestamp has a fixed large value. We attempted to use other characteristics of the labeled anomalies to find the exact timestamp. We particularly used the 4-tuples information (source IPs, source ports, destination IPs and destination ports) contained in the labels to find the network flows that match. Since labels can contain a 4-tuple that has more than one property (e.g., source IP) that is not specified, we often found too many matched network flows for an anomaly. As a result, most of the time windows are labeled as anomalous in our training dataset. We found that the labeled dataset obtained using that method was not separable enough for network attack detection and classification. Therefore, we leverage synthetic attack injection tool in order to obtain a good labeled dataset.

3. Methods

3.1. Data processing pipeline

Figure 4 shows the outline of the data processing pipeline from the raw pcap network traffic to the input features of the Machine Learning model. The real network traffic in MAWI pcap files are injected with different types of synthetic attacks and levels of intensity using a tool called ID2T [13] [14]. We use YAF [23] [24], a network flow extractor, to extract network flow information from these injected files. In order to convert the binary file generated by YAF to a tabular format, we use YAFSCII [25]. In the feature extraction phase, these network flows are placed into 10-second time windows and the entropies of the features of interest are computed. The Machine Learning classification model uses these entropies as input features to learn and to predict network attacks. Next sections will describe these processing steps in more details.

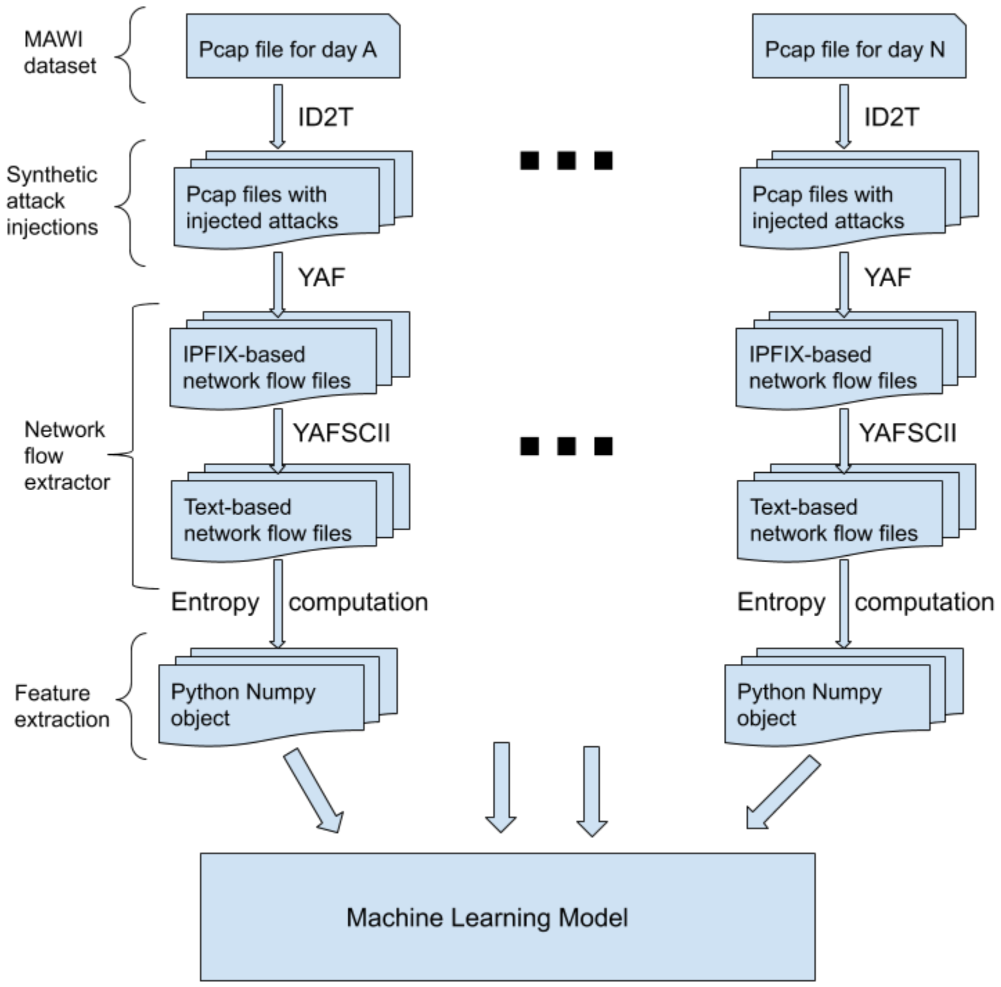


Figure 4. Data processing pipeline: We inject synthetic attacks to the real network traffic on the MAWI dataset using ID2T, extract network flows from the dataset using YAF, convert the files to a tabular format using YAFSCII, and compute Shannon entropies of network flow features. We use these features to train a deep neural network to detect and classify network attacks.

3.2. Dataset

The MAWI dataset is provided by the WIDE Project and publicly available on the data repository maintained by the MAWI Working Group [15]. Specifically, this project uses sample point F, containing daily packet traces at the transit link of WIDE to the upstream ISP. Each day from 2:00pm to 2:15pm (900 seconds total), raw network traffic going through the WIDE Internet backbone link was captured and archived [15]. These recorded packets contain only packet headers with no application payload for privacy concern.

This thesis uses the pcap traffic data from two separate months, March 2015 and June 2019, which were arbitrarily chosen. The dataset is of substantial size. During a period of 15 minutes in one single day, the size of the raw pcap files is from 6GB to 13GB, which contains only protocol headers and no payloads, and the number of captured packets is in the range of 70 million to over 200 million. The traffic rate has an average of approximately 160,000 packets per second. The total captured size (header alone) is about 10GB in 900 seconds. This highlights a huge amount of traffic going through the Internet backbone link.

3.3. Synthetic attack injection

Given the difficulty in determining the ground truth from the MAWI dataset, we leverage a synthetic data generator, Intrusion Detection Dataset Toolkit (ID2T) [13] [14], to inject various attacks into the network traffic. ID2T is a research tool intended to assist cybersecurity researchers in creating labeled datasets with user-defined synthetic attacks [13]. ID2T is an ongoing project, and is continuously evolving [14].

Four types of attacks are injected: SYN Flood DDoS attack, port scan, point-to-point DoS attack, and network scan. Different intensity levels of attacks are injected to the original dataset to train the model and to measure the model performances. On each day, attacks of different types and different intensities are injected to 9 different points on the original dataset. More information about each type of injected attack is discussed further in the section 4.

Using the methods and the Replicator Neural Network as described in [9], we plotted the Anomaly Scores of the traffic in one sample day (June 30, 2019). In Figure 5, Anomaly Scores of the original traffic are shown on the top, and the Anomaly Scores of the traffic injected with DDoS attacks of 10K packets/second are shown on the bottom.

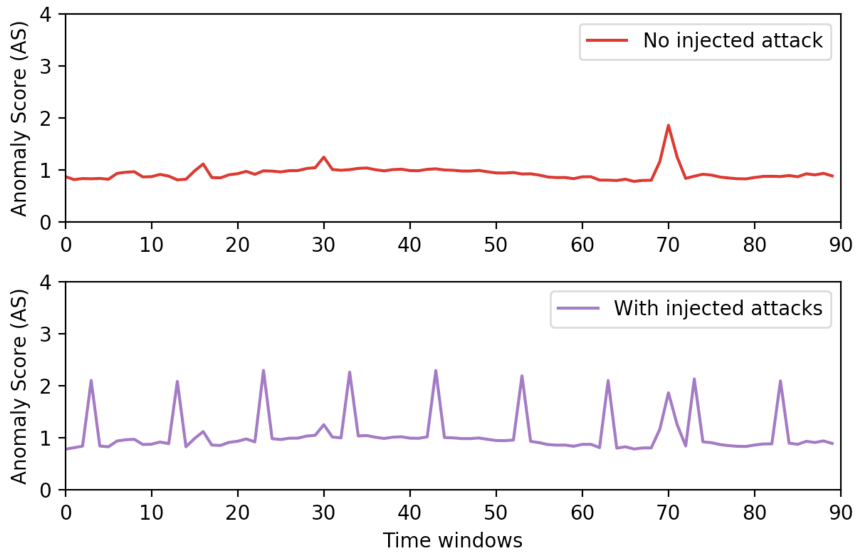
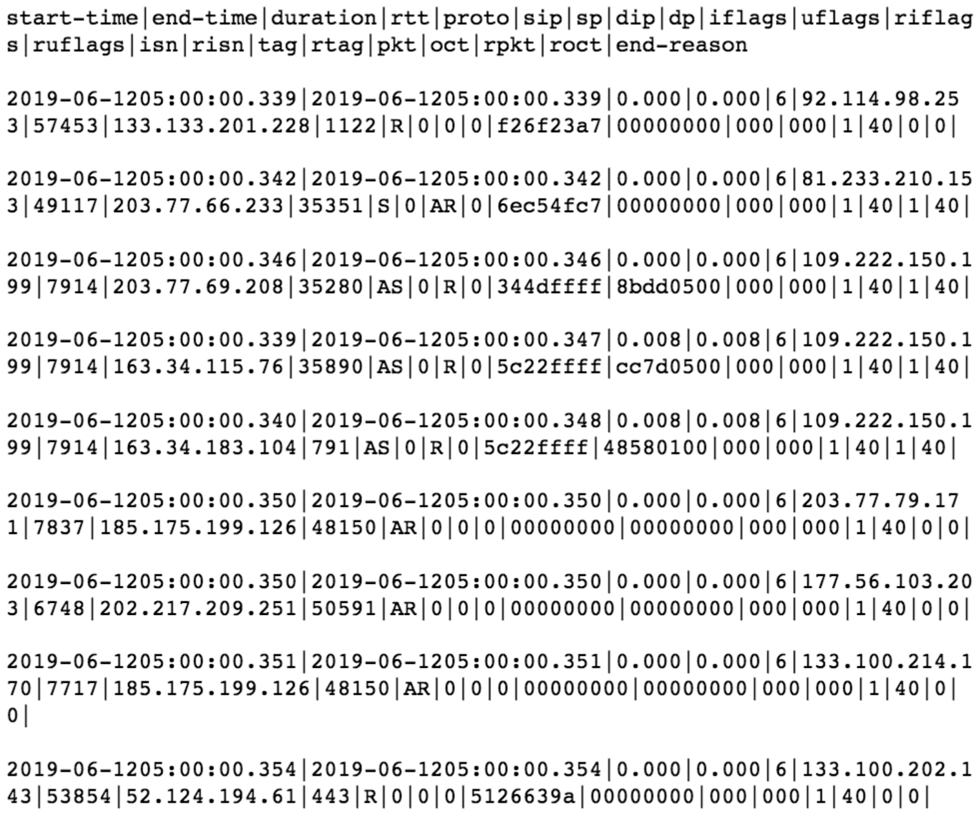


Figure 5. Anomaly Scores for the original traffic and traffic with injected attacks: We injected synthetic attacks at 9 different time windows in each day traffic. The plot on the top shows Anomaly Scores of real traffic in the MAWI dataset on the day 2019-06-30. The plot on the bottom shows the Anomaly Score changes at the time windows where DDoS attacks of 10K packets/second are injected

3.4. Network flow extractor

A network flow is a sequence of packets from a source to a destination. RFC 2722 describes flows as “an artificial logical equivalent to a call or connection” [26]. To extract network flows from raw pcap files, we use YAF (Yet Another Flowmeter) [27]. This tool processes raw packet data and exports the network flows to an IPFIX-based file format [28]. YAFSCII is a related tool that takes the IPFIX-based files generated by YAF and generates human-readable ASCII text files in a tabular format with one flow per line [25]. Figure 6 shows the content and the format of a YAFSCII file. It includes many network flow properties such as start time, end time, duration, round trip time, IP protocol, source IP, source port, destination IP and destination port. The entropies of these network flow properties are used as input features to the classification model.



```
start-time|end-time|duration|rtt|proto|sip|sp|dip|dp|iflags|uflags|riflag
s|ruflags|isn|risn|tag|rtag|pkt|oct|rpkt|roct|end-reason

2019-06-1205:00:00.339|2019-06-1205:00:00.339|0.000|0.000|6|92.114.98.25
3|57453|133.133.201.228|1122|R|0|0|0|f26f23a7|00000000|000|000|1|40|0|0|

2019-06-1205:00:00.342|2019-06-1205:00:00.342|0.000|0.000|6|81.233.210.15
3|49117|203.77.66.233|35351|S|0|AR|0|6ec54fc7|00000000|000|000|1|40|1|40|

2019-06-1205:00:00.346|2019-06-1205:00:00.346|0.000|0.000|6|109.222.150.1
99|7914|203.77.69.208|35280|AS|0|R|0|344dffff|8bdd0500|000|000|1|40|1|40|

2019-06-1205:00:00.339|2019-06-1205:00:00.347|0.008|0.008|6|109.222.150.1
99|7914|163.34.115.76|35890|AS|0|R|0|5c22ffff|cc7d0500|000|000|1|40|1|40|

2019-06-1205:00:00.340|2019-06-1205:00:00.348|0.008|0.008|6|109.222.150.1
99|7914|163.34.183.104|791|AS|0|R|0|5c22ffff|48580100|000|000|1|40|1|40|

2019-06-1205:00:00.350|2019-06-1205:00:00.350|0.000|0.000|6|203.77.79.17
1|7837|185.175.199.126|48150|AR|0|0|0|00000000|00000000|000|000|1|40|0|0|

2019-06-1205:00:00.350|2019-06-1205:00:00.350|0.000|0.000|6|177.56.103.20
3|6748|202.217.209.251|50591|AR|0|0|0|00000000|00000000|000|000|1|40|0|0|

2019-06-1205:00:00.351|2019-06-1205:00:00.351|0.000|0.000|6|133.100.214.1
70|7717|185.175.199.126|48150|AR|0|0|0|00000000|00000000|000|000|1|40|0|
0|

2019-06-1205:00:00.354|2019-06-1205:00:00.354|0.000|0.000|6|133.100.202.1
43|53854|52.124.194.61|443|R|0|0|0|5126639a|00000000|000|000|1|40|0|0|
```

Figure 6. YAFSCII file in tabular format: This figure shows the content of a YAFSCII file, which contain properties of workflow, such as start time, end time, duration, round trip time, IP protocol, source IP, source port, destination IP, destination port, and so on delimited by the "|" character. More information on these properties can be found in the YAFSCII Linux man page [25].

In this experiment, we set the active timeout and idle timeout to 10 seconds. These parameters are the same as those in the previous experiment [9] and are consistent with the time window parameter in the feature extraction phase.

3.5. Feature extraction

The network flow files produced by YAFSCII are parsed and the flows are segmented into 10-second time windows. For each time window, Shannon entropies associated with each of the 14 features of interests (e.g. source IP, source port, destination IP and destination port) are computed. Entropy is a measure of uncertainty associated with a random variable and is used extensively in information theory. Shannon entropy for each feature is calculated as:

$$\boldsymbol{H = -\sum p_i log(p_i)}$$

where p_i is the probability a feature instance.

Table 1 lists the 14 features of interest that are used to compute Shannon entropies. These Shannon entropies will be used as input features to the Machine Learning classification model. As shown in the table, these features of interest also contain combinations of two network flow features, such as Destination IP – Destination Port because the entropies of these combination features are beneficial in classifying attacks of different types.

The feature extraction phase is implemented in Python. It outputs the Shannon entropies as numpy objects which are saved in files for further analysis.

Features of interest	**Explanation**
Source IP	Source IP address
Source Port	Source port
Destination IP	Destination IP address
Destination Port	Destination port
Protocol	IP protocol
Initial Flags	Forward first-packet TCP flags
Union Flags	Forward nth-packet TCP flags union
Reverse Initial Flags	Reverse first-packet TCP flags
Reverse Union Flags	Reverse nth-packet TCP flags union
End reason	Indicate whether the flow was ended normally (i.e., by TCP RST or FIN), expired by idle timeout, or expired by active timeout.
Destination IP – Destination Port	Combination of Destination IP and Destination Port
Destination IP – Initial Flags	Combination of Destination IP and Initial Flags
Source IP - Destination IP	Combination of Source IP and Destination IP
Source IP - Initial Flags	Combination of Source IP and Initial Flags

Table 1. Features of interest from network flows: This table lists network flow features and combinations of network flow features that are used to compute Shannon entropies. The deep neural network uses these entropies as input features to predict and classify network attacks.

4. Synthetic Attack injections

This thesis focuses on detecting and classifying 4 types of network attacks: Distributed Denial of Service, port scan, point-to-point Denial of Service, and network scan. Further work can be extended to other types of attacks. In this section, we discuss the attacks generated by ID2T and the parameters used to generate them. ID2T is still evolving at the time of this experiment. As such, its behavior might change in the future.

4.1. Distributed Denial of Service (DDoS) Attack

In a DDoS attack, ID2T generates a large number of SYN packets which have different source IP addresses and source ports to attack one single target. Five different levels of intensity are used to generate SYN DDoS attacks. Table 2 shows the ID2T parameters and calculated values at different DDoS attack intensities. The *packets.per-second* parameter indicates the number of packets sent per second by attackers. The *attack.duration* parameter is the duration of the attack and is set to 5 seconds. The *attackers.count* parameter is the number of attackers and is specified to ensure that different source IP addresses and ports are generated.

The number of packets generated can be found in the output label file after an ID2T process is finished. It varies depending on attack types and attack parameters. In DDoS attack type, ID2T also generates response packets sent by victims. Thus, this value is equal to 2 × *packets.per-second* × *attack.duration*. The generated packet rate is calculated by dividing the number of packets generated to the attack duration. The "percentage" column shows the ratio between the rate of generated packets and the average packet rate in the original traffic, which is approximately 160,000 packets per second.

packets. per-second (param)	**attack. duration (param)**	**attackers. count (param)**	**Number of packets generated**	**Generated packet rate**	**Percentage**
20,000	5	50,000	200,000	40,000	25%
10,000	5	25,000	100,000	20,000	12.5%
5,000	5	12,500	50,000	10,000	6.25%
3,000	5	7,500	30,000	6,000	3.75%
1,000	5	2,500	10,000	2,000	1.25%

Table 2. ID2T parameters for DDoS attacks at different intensities: This table lists the parameters and calculated values for different intensities of DDoS attack.

4.2. Port Scan Attack

In port scan attack, an attacker sent packets to multiple ports on a single IP addresses. Five different levels of intensity were used to generate port scan attacks. The *packets.per-second* parameter specifies the number of packets sent per second by the attacker. The *port.dst* parameter specifies the ports to be scanned. There is no *attack.duration* parameter in the port scan attack. Therefore, the *packets.per-second* and *port.dst* parameters must be set accordingly to ensure that the duration of the scan is 5 seconds. Table 3 shows the ID2T parameters and calculated values at different port scan attack intensities.

Unlike the DDoS attack, ID2T does not generate response packets for port scan. Therefore, the number of packets generated is equal to the number of destination ports. The rate of generated packets and the percentage are calculated similar to those of DDoS.

packets. per-second (param)	**port.dst (param)**	**Number of packets generated**	**Generated packet rate**	**Percentage**
13,107	1..65535	65,535	13,107	8.2%
9,830	1..49151	49,151	9,830	6.1%
6,000	1..30000	30,000	6,000	3.75%
2,000	1..10000	10,000	2,000	1.25%
205	1..1024	1,024	205	0.13%

Table 3. ID2T parameters for Port scan attacks at different intensities: This table lists the parameters and calculated values for different intensities of port scan attack.

4.3. Point-to-Point Denial of Service (DoS) Attack

In a point-to-point denial of service attack, ID2T generates a large number of SYN packets with the same source IP address and different ports to attack one single target. Four intensity levels were used to generate the Point-to-Point Denial of Service attacks. These attacks are generated using the same ID2T type of attack (DDoSAttack) as in section 4.1. To ensure that there is only one attacker, the *attackers.count* parameter is set to 1. Table 4 shows the ID2T parameters and calculated values at different intensities of point-to-point DoS attack. Similar to a DDoS attack, ID2T generates response packets sent by victims.

packets. per-second (param)	attack. duration (param)	attackers. count (param)	Number of packets generated	Generated packet rate	Percentage
10,000	5	1	100,000	20,000	12.5%
5,000	5	1	50,000	10,000	6.25%
3,000	5	1	30,000	6,000	3.75%
1,000	5	1	10,000	2,000	1.25%

Table 4. ID2T parameters for point-to-point DoS attacks at different intensities: This table lists the parameters and calculated values for different intensities of point-to-point DoS attack.

4.4. Network Scan

Since ID2T currently does not have a dedicated network scan attack type, we use SMB Scan as an example of network scan. In a SMB scan attack, attackers scan a network for SMB (Server Message Block) servers on port 445. If a TCP connection is successfully established, an SMB negotiation starts and more packets are exchanged. Four different levels of intensity were used to generate the SMB Scan attacks as shown in Table 5. The *target.count* parameter specifies the number of targets to be scanned. The number of targets that actually host a SMB server is 50% by default. The *packets.per-second* parameter does not specify the number of packets per second as its name suggests, rather, it implies the number of targets per second. At the time of this writing, the ID2T authors are aware of this oversight, thus it will be fixed soon. The *target.count* and *packets.per-second* (more accurately, *targets.per-second*) parameters are set such that the entire attack lasts 5 seconds.

packets. per-second (param)	**target. count (param)**	**Number of packets generated**	**Generated packet rate**	**Percentage**
3,000	15,000	90,000	18,000	11.25%
2,000	10,000	60,000	12,000	7.5%
1,000	5,000	30,000	6,000	3.75%
300	1,500	9,000	1,800	1.13%

Table 5. ID2T parameters for SMB Network scan attacks at different intensities: This table lists the parameters and calculated values for different intensities of SMB network scan attack.

5. Machine Learning model

5.1. Training dataset and Validation Dataset

A set of traffic data in 36 days (from 03/09/2015 to 03/22/2015, and from 06/01/2019 to 06/22/2019) are used for training. A second set of traffic data in 17 days (from 03/23/2015 to 03/31/2015, and from 06/23/2019 to 06/30/2019) are used for validation.

For each day, 18 injected files were generated with different attack types and attack intensities. They are DDoS (20K, 10K, 5K, 3K and 1K packets/second), Port Scan (13K, 9.8K, 6K, 2K, 205 packets/second), Point to Point DoS (10K, 5K, 3K, 1K packets/second) and Network Scan (3K, 2K, 1K, 300 targets/second). The type of attacks and intensities were discussed in details in section 4. In each injected file, 9 attacks were injected at 9 separate time windows. Therefore, for each type of attacks and intensities, there are 324 training instances and 153 validation instances.

5.2. Input Features

The input features for the Machine Learning classification model are 14 Shannon entropies of the 14 features of interest of network flows as shown in the Table 1. More detailed information about the network flow features can be found at the Linux man page of YAFSCII [25].

It is possible that a few of the interested features has no or little benefit on attack detection and classification. The Machine Learning model, however, is capable of ignoring the irrelevant features by adjusting the weight parameters of those features to an insignificant value in the training phase. A possible future work is to refine input features and explore the new ones to improve the model accuracy.

5.3. Classification Model

The classification model is a fully-connected feedforward neural network consisting of an input layer, 3 hidden layers, and an output layer. The input layer takes in 14 input features as described above. The 3 hidden layers have 100, 30 and 100 nodes, respectively. All three hidden layers use *rectified linear unit* (ReLU) as the activation function. The output layer uses *softmax* as the activation function. It consists of 5 categorical classes: No-Attack, DDoS-Attack, Port-Scan-Attack, Point2Point-DoS-Attack, and Network-Scan-Attack. The classification model is implemented in Python using Tensorflow.

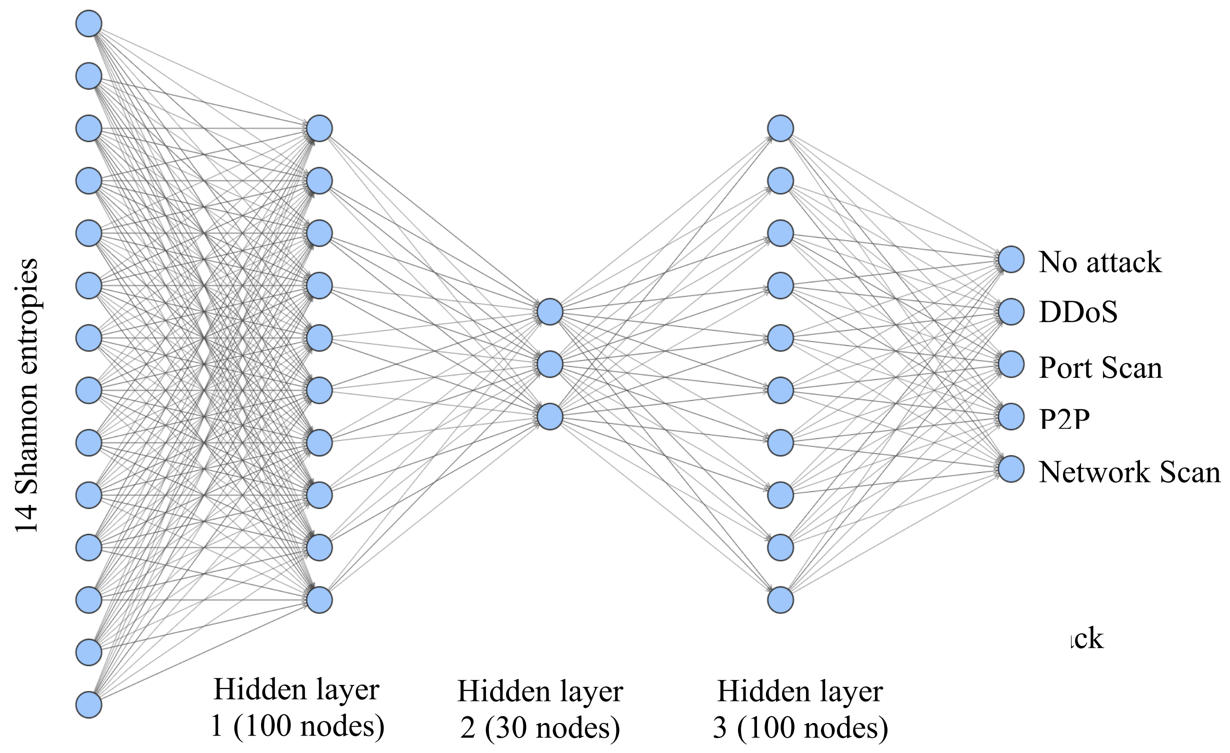

14 Shannon entropies
Hidden layer 1 (100 nodes)
Hidden layer 2 (30 nodes)
Hidden layer 3 (100 nodes)
No attack
DDoS
Port Scan
P2P
Network Scan
ck

Figure 7. The Deep Neural Network for attack detection and classification: The classification model is a Fully Connected Neural Network consists of an input layer (14 nodes correspond to 14 features), 3 hidden layers (100 nodes, 30 nodes, and 100 nodes), and an output layer (5 nodes correspond to 5 categories). Because of limited space, not all nodes in the hidden layers can be drawn. In the figure, one node in the hidden layers represents 10 nodes in the model.

6. Evaluation

The performance of the classification model on the validation dataset is shown in the heat map in Figure 8. The vertical axis shows the attack labels, and the horizontal axis shows the model prediction. This confusion matrix is normalized such that the sum of each row is added up to 1. Values in the diagonal reflect the accuracies of each type of attack.

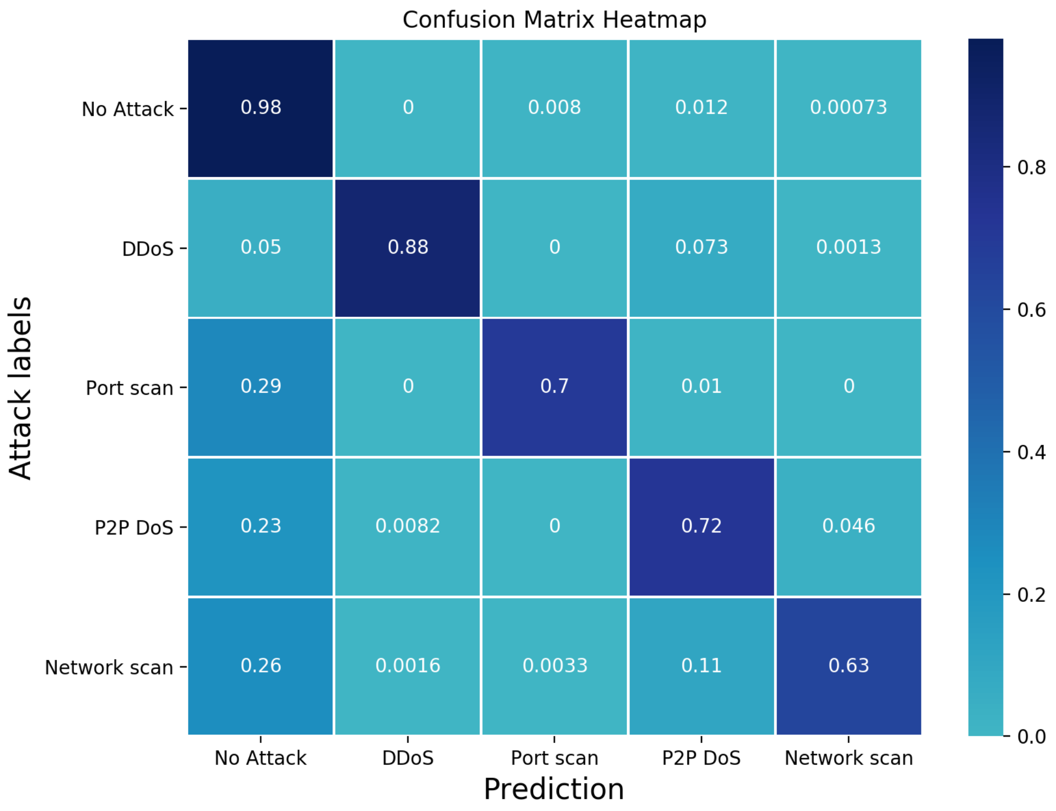


Figure 8. Confusion matrix for validation data: This matrix summarizes how the deep learning model performs on all levels of attack intensity. The vertical axis shows the ground truth, and the horizontal axis shows the model prediction.

The matrix summarizes the model performances on all intensity levels of an attack. It is worth to mention that low intensity attacks are unlikely to cause noticeable changes in the entropies of any of the interested network flow features.

When considered only high and moderate intensity attacks, the classification model achieves a high level of accuracy. The confusion matrix in Figure 9 shows the classification performance for attacks with intensity at least 5,000 packets per second (for DDoS, Port scan, P2P DoS attacks) or 2,000 targets per second (for network scan attacks). To put it in context, the rate of traffic in the Internet backbone link is about 120,000 to 200,000 packets per second.

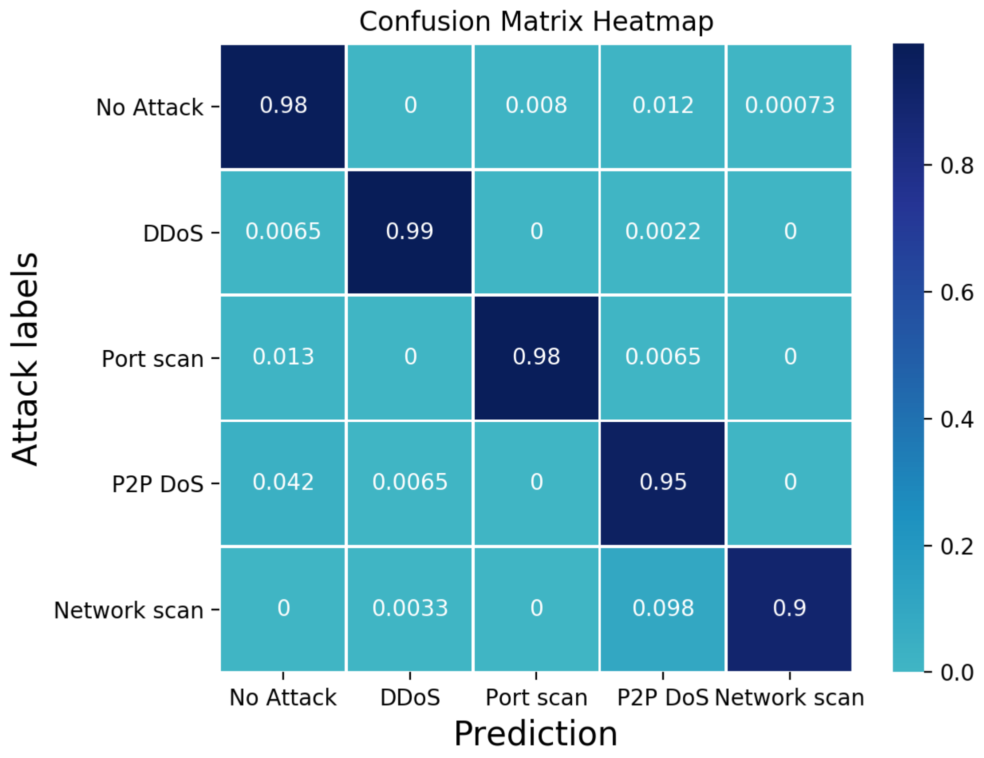
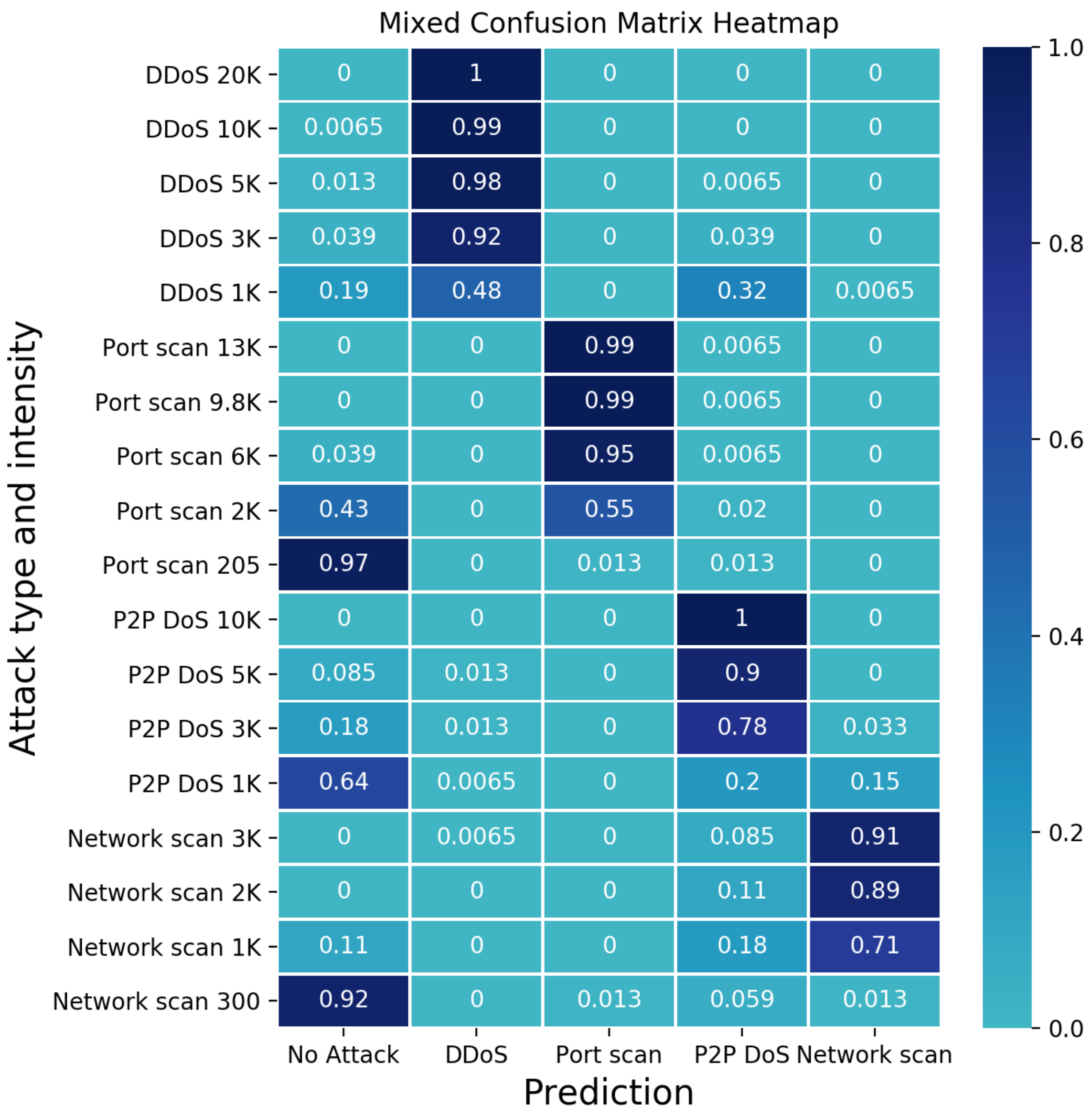


Figure 9. Confusion matrix for validation data injected with high and moderate intensity attacks: This matrix summarizes how the deep learning model performs on high and moderate intensity attacks. The vertical axis shows the ground truth, and the horizontal axis shows the model prediction.

Figure 10 provides additional information about the model performance on different types of attacks and intensities. As indicated in this figure, the model is likely to predict a low intensity attack as no attack and it is unlikely to mistake one type of attack for another.



	No Attack	DDoS	Port scan	P2P DoS	Network scan
DDoS 20K	0	1	0	0	0
DDoS 10K	0.0065	0.99	0	0	0
DDoS 5K	0.013	0.98	0	0.0065	0
DDoS 3K	0.039	0.92	0	0.039	0
DDoS 1K	0.19	0.48	0	0.32	0.0065
Port scan 13K	0	0	0.99	0.0065	0
Port scan 9.8K	0	0	0.99	0.0065	0
Port scan 6K	0.039	0	0.95	0.0065	0
Port scan 2K	0.43	0	0.55	0.02	0
Port scan 205	0.97	0	0.013	0.013	0
P2P DoS 10K	0	0	0	1	0
P2P DoS 5K	0.085	0.013	0	0.9	0
P2P DoS 3K	0.18	0.013	0	0.78	0.033
P2P DoS 1K	0.64	0.0065	0	0.2	0.15
Network scan 3K	0	0.0065	0	0.085	0.91
Network scan 2K	0	0	0	0.11	0.89
Network scan 1K	0.11	0	0	0.18	0.71
Network scan 300	0.92	0	0.013	0.059	0.013

Figure 10. Mixed confusion matrix for different types of attack and intensity: This matrix gives detailed information about the model performances on different types of attack and intensity. The vertical axis shows the ground truth, and the horizontal axis shows the model prediction.

To further evaluate the model performance on different types of attack and intensity, we measured and plotted the model accuracy against the intensity of an attack. The results are shown in Figure 11, Figure 12, Figure 13 and Figure 14. All of the graphs suggest a strong positive correlation between attack intensity and model accuracy.

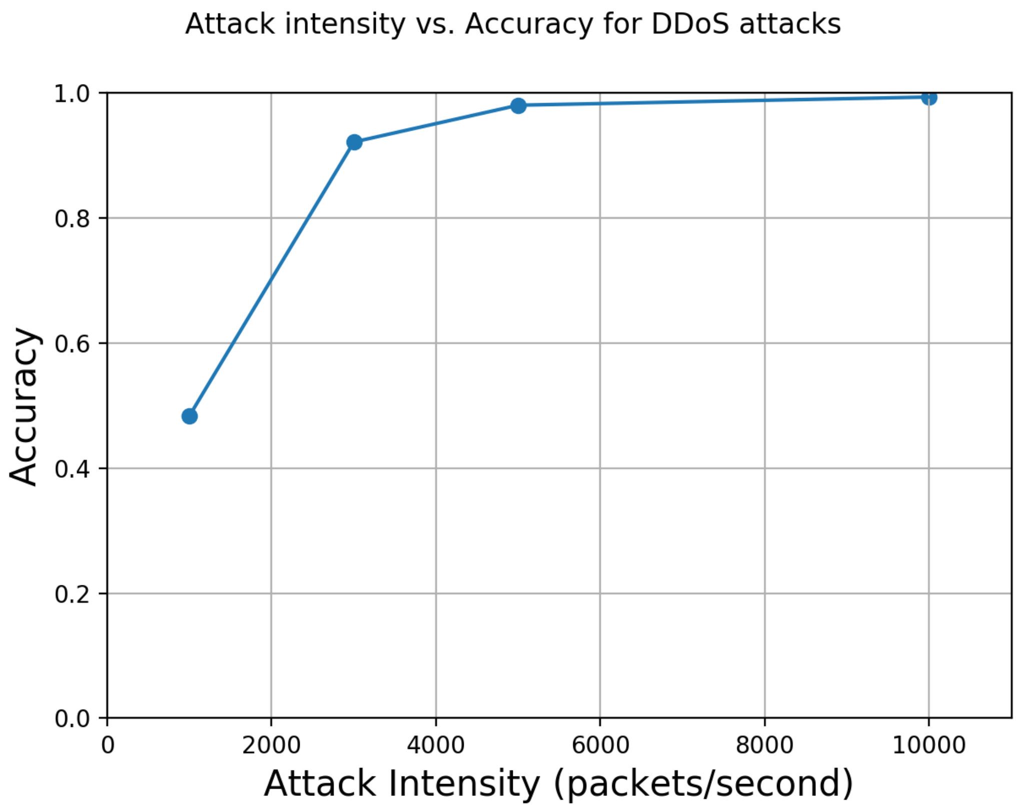


Figure 11. Attack intensity vs. Accuracy for DDoS attacks: This plot shows the model accuracy at different intensities of DDoS attack. The attack intensity is the number of packets sent by the attackers per second.

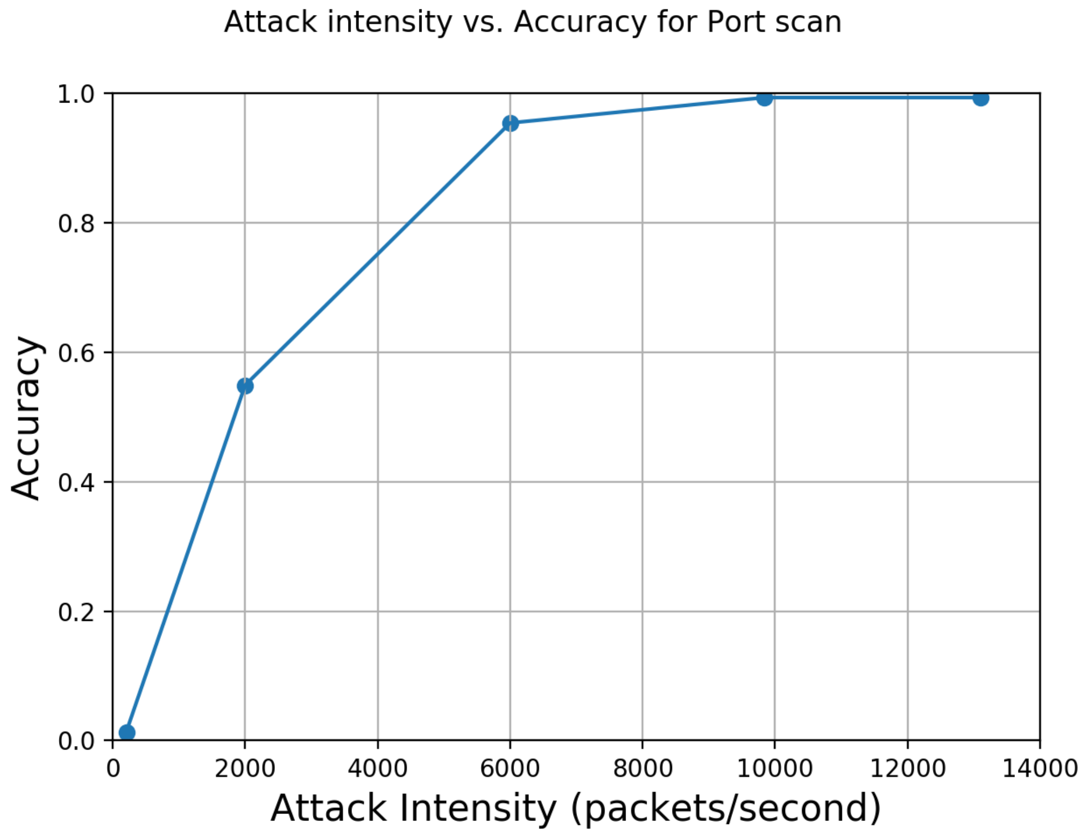


Figure 12. Attack intensity vs. Accuracy for Port scan attacks: This plot shows the model accuracy at different intensities of Port scan attack. The attack intensity is the number of packets sent by the attackers per second.

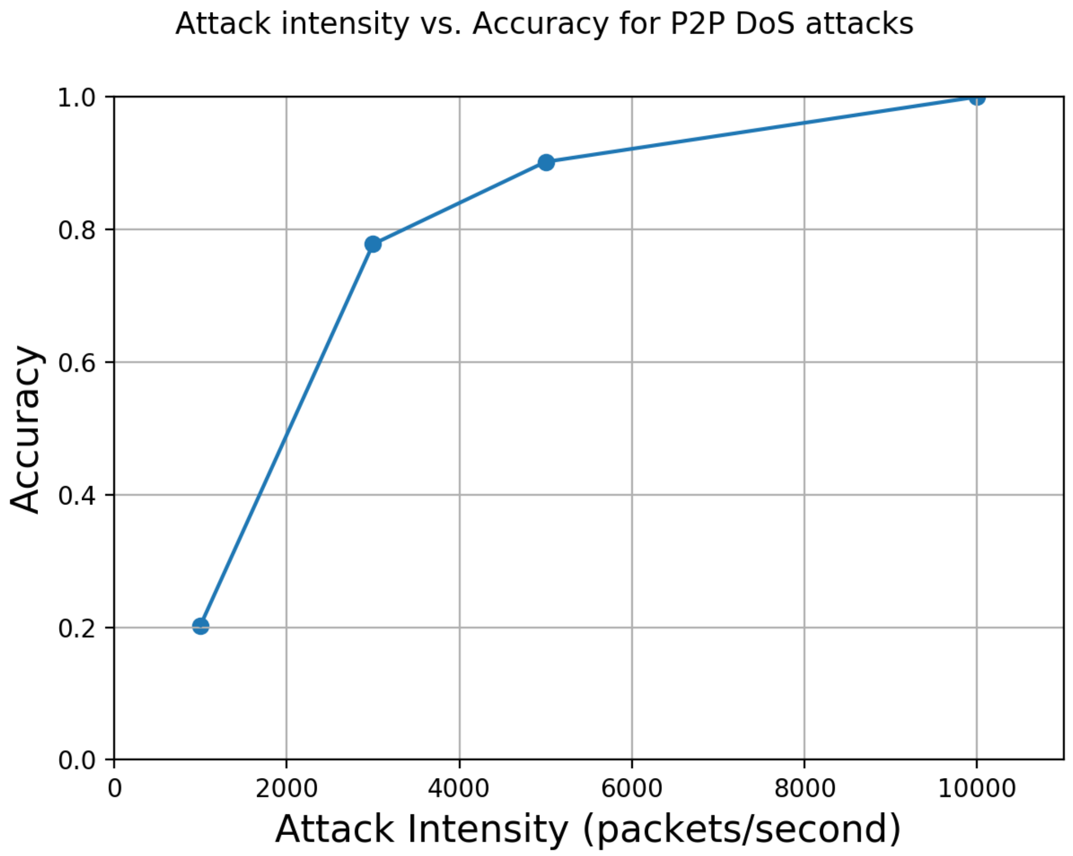


Figure 13. Attack intensity vs. Accuracy for point-to-point DoS attacks: This plot shows the model accuracy at different intensities of point-to-point DoS attack. The attack intensity is the number of packets sent by the attackers per second.

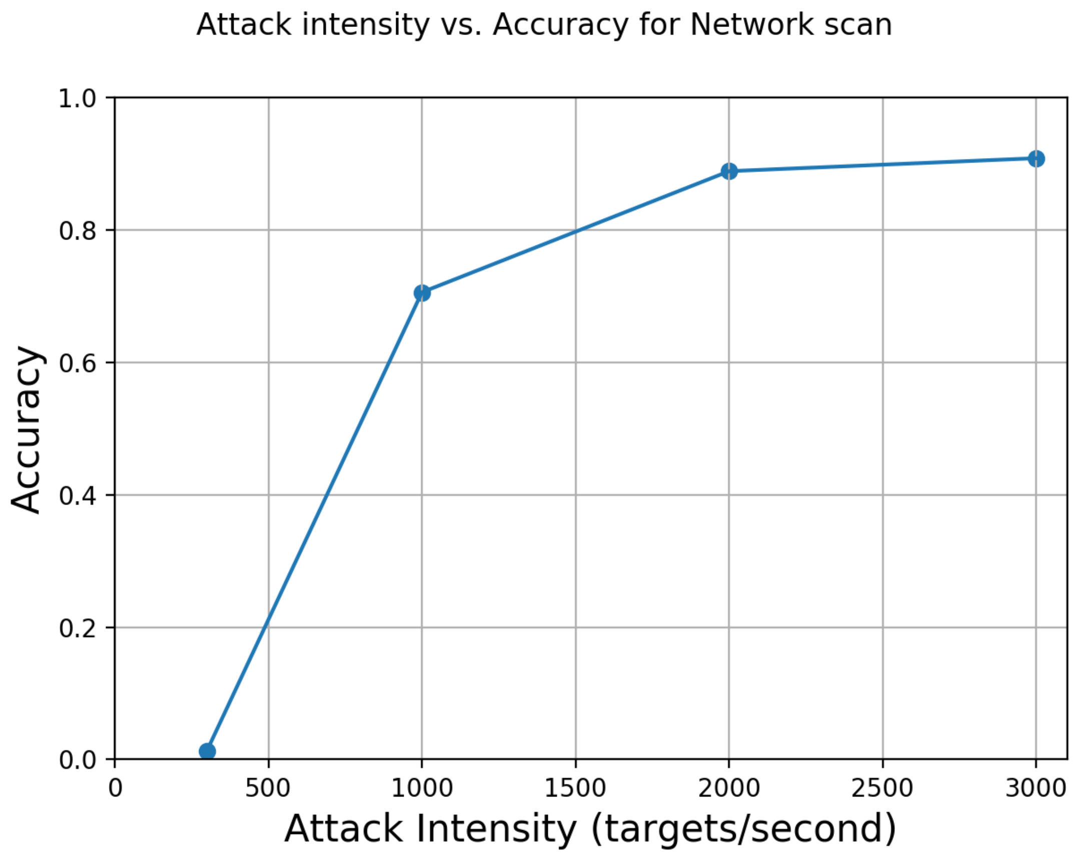


Figure 14. Attack intensity vs. Accuracy for Network scan attacks: This plot shows the model accuracy at different intensities of SMB Network scan attack. The attack intensity is the number of targets per second.

It is important to note that the units in the horizontal axes do not have the same meanings because ID2T works differently for different types of attack. In the DDoS and point-to-point DoS attacks, the horizontal axes show the packet rates sent by attackers, but ID2T also injects the response packets sent by the victims, thus the injected packet rates for these two attacks are doubled. In the port scan attack, ID2T does not inject the response packets sent by the victims, therefore, the injected packet rate is the same as the rate shown in the horizontal axis. In the SMB network scan attack, the horizontal axis shows the number of targets per second (even though the parameter is named *packets.per-second*) and the number of packets generated is 6 times the number of targets. To compare the performances of these attacks to each other, we combine them into one single figure using the percentage of injected traffic as the common unit. As mentioned in the section 4, these percentages are calculated using 160,000 as the average packet rate in the original traffic.

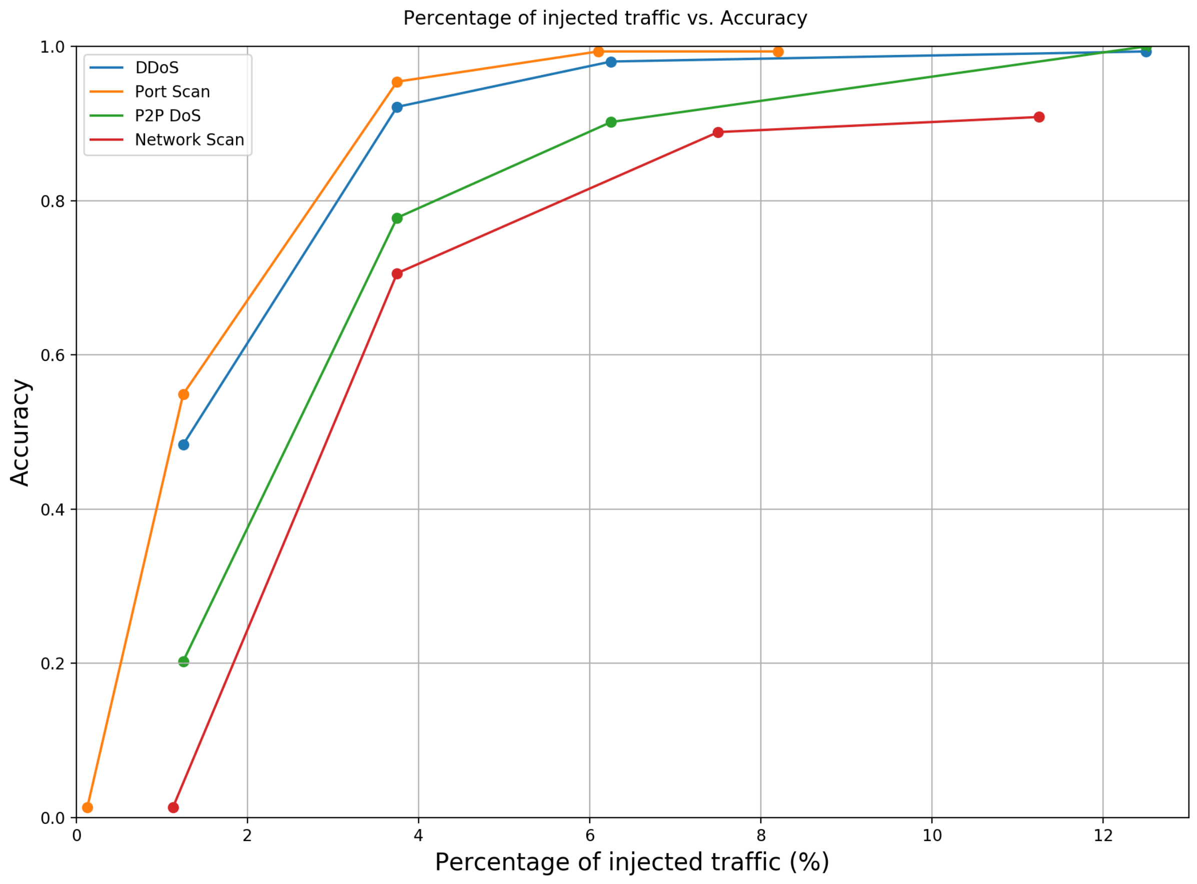


Figure 15. Percentage of injected traffic vs. Accuracy: This figure compares the model accuracy of 4 different types of attack. The horizontal axis indicates the ratio of injected packet rate to the average packet rate of real traffic (160,000 packets/second).

Figure 15 compares the performances of the 4 types of attack. Interestingly, they have very similar shapes. The port scan attack has the best performance, probably because its injected traffic does not include response packets sent by victims. Even though the DDoS and point-to-point DoS injections work similarly, the DDoS performs better, possibly because it causes greater changes in entropy of certain features of interest. The lowest accuracy rate in the network scan is likely because ID2T generates 6 packets per target on average for this type of attack.

Moreover, Figure 15 indicates that the model accuracy increases rapidly as the injected traffic increases from 1% to 4%. The accuracy increases less when the injected traffic is from 4% to 6% and starts level out to reach maximum after that point. It also confirms that the model achieves a high accuracy when the amount of injected traffic is as low as 6% of the original traffic. The results confirm that entropies of network flow are promising features in detecting and classifying network attacks.

7. Computational Performance

7.1. Synthetic attack injection time

The ID2T processes for injecting synthetic attacks were run as submitted jobs on the MIT SuperCloud [29] [24] [23], a powerful computing platform capable of supporting research projects that required significant amount of computing power, memory and big data resources [29]. Figure 16 shows the typical time taken to inject each type of attack and intensity into one day traffic using ID2T in this computing platform. The total time is approximately 6 hours to process one day traffic.

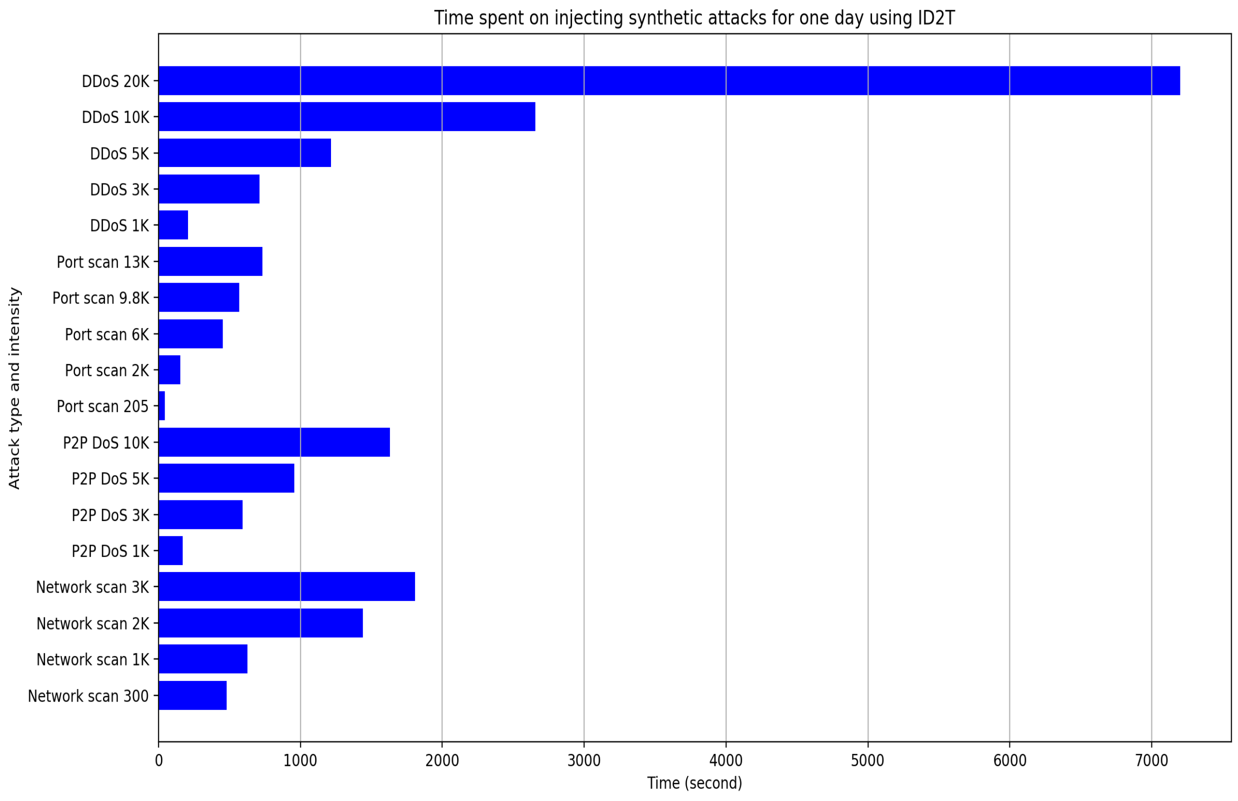


Figure 16. Synthetic attack injection time with ID2T: The figure shows the average time to inject different types of attack and intensity to one day traffic. It is the time for ID2T to inject attacks to 9 different time windows. The high intensity attacks take longer time to inject than the low intensity ones.

7.2. Pre-processing time

The pre-processing time includes the time for extracting network flows from the raw pcap files and computing entropies for interested features. It excludes the synthetic attack injection time in the previous section. The pre-processing code was written in Python and run as submitted jobs on the MIT SuperCloud [29] [24] [23]. Figure 17 displays the average time it took to process a MAWI pcap file injected with one type of attack and intensity, which approximates to 40 minutes in total.

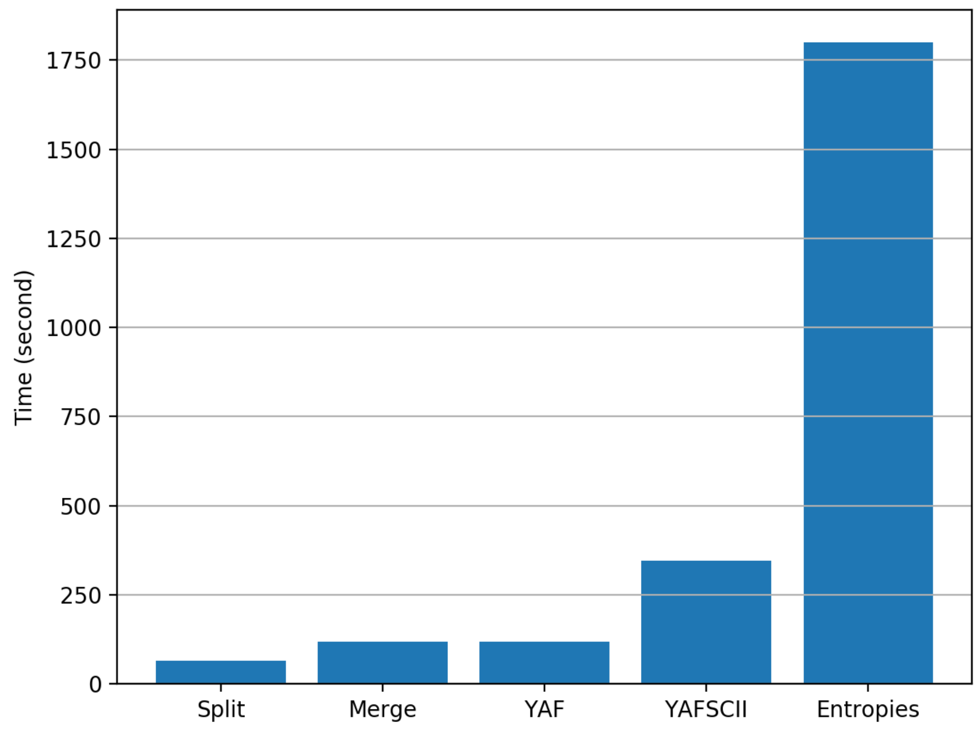


Figure 17. Pre-processing time: The figure shows the average time to process a MAWI pcap file injected with one type of attack and intensity. The total time is approximately 40 minutes.

The split and merge steps have a sole purpose of speeding up the synthetic attack injection processes. It would take a long time to inject synthetic attacks directly to big input pcap files. Thus, we split input files into smaller ones, and inject synthetic attacks to one of the split files, then remerge the files back together.

As shown in Figure 17, the majority of the time is spent on the “entropies” step. In fact, this “entropies” time includes the time for parsing the YAFSCII text files, placing network flows into time windows and computing entropies for the 14 network flow features. A large portion of the time, however, is spent on I/O and parsing text files, as the size of the text files is 12GB big on average. We believe it would take much less of the time if we parsed YAF files directly from their IPFIX binary format without converting them to a tabular format as an intermediate step, as the file size in the IPFIX format is typically 5 times smaller than its text form. Nevertheless, since a binary parser for YAF files is not readily available and this step is not a critical path of the data analysis, we did not attempt to optimize it in terms of computational performance.

7.3. Training time and prediction time

The time to train the deep neural network was approximately 3 minutes to run 2000 epochs to achieve 97% accuracy on the training dataset and 95% on the validation dataset. The prediction step is very quick and can be considered as real time.

8. Conclusions and Future work

Building a deep neural network to detect and classify network attacks is the core of this thesis. Our approach based on the hypothesis that each type of network attacks generates a distinguishable change in the entropies of network flow features. Experimental results show that our machine learning model can detect and classify high and moderate intensity network attacks with high accuracy. The result is very promising, considering the fact that the amount of traffic in the Internet backbone link is substantial but the model can still detect and classify attacks that constitute as low as 6% of the original traffic. The experimental results confirm that entropies of network flow are promising features in detecting and classifying network attacks. This work also quantifies the model performance on different attack types and intensities. Interestingly, different types of attack show similar shapes when plotting the model accuracy against the percentage of injected traffic.

Besides other constraints, this experiment is restricted by the current limited capability of ID2T and the fact that the MAWI dataset does not contain any payload. For future work, this thesis can be expanded to more types of network attacks and to other types of network architectures, such as a LAN network. We believe that this method would perform even better in a LAN network because the amount of traffic in most LAN networks is significantly lower than that of an Internet backbone link. Therefore, even low intensity attacks could generate enough entropy changes for the model to detect and as a result, the model could detect and classify these low intensity attacks with high accuracy. In addition, the availability of packet payloads would open up a number of additional features that could be utilized by machine learning models for detecting and classifying more types of network attacks, especially the types of attacks that could only be identified by the application payloads.

Exploring new input features can also be performed. As an example, instead of Shannon entropies, other types of entropies such as parameterized entropies [12] may yield better results. Finally, there is a potential of improving the detection and classification accuracy by using other types of deep learning techniques.

Bibliography

[1] Y. LeCun, Y. Bengio and G. Hinton, "Deep learning," *Nature,* 28 May 2015.

[2] V. Chandola, A. Banerjee and V. Kumar, "Anomaly Detection : A Survey," 2009. [Online]. Available: http://cucis.ece.northwestern.edu/projects/DMS/publications/AnomalyDetection.pdf.

[3] R. K. Malaiya, D. Kwon, J. Kim, S. C. Suh, H. Kim and I. Kim, "An Empirical Evaluation of Deep Learning for Network Anomaly Detection," 2018. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8390278.

[4] D. Kwon, K. Natarajan, S. C. Suh, H. Kim and J. Kim, "An Empirical Study on Network Anomaly Detection Using Convolutional Neural Networks," 2018. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8416441.

[5] W.-H. Lin, H.-C. Lin, P. Wang, B.-H. Wu and J.-Y. Tsai, "Using convolutional neural networks to network intrusion detection for cyber threats," 2018. [Online]. Available: https://ieeexplore.ieee.org/document/8394474.

[6] R. Vinayakumar, K. P. Soman and P. Poornachandran, "Applying convolutional neural network for network intrusion detection," 2017. [Online]. Available: https://ieeexplore.ieee.org/document/8126009.

[7] S. Naseer, Y. Saleem, S. Khalid, M. K. Bashir, J. Han, M. M. Iqbal and K. Han, "Enhanced Network Anomaly Detection Based on Deep Neural Networks," 2018. [Online]. Available: https://ieeexplore.ieee.org/document/8438865.

[8] S. Hawkins, H. He, G. Williams and R. Baxter, "Outlier Detection Using Replicator Neural Networks," in *International Conference on Data Warehousing and Knowledge Discovery*, Springer, Berlin, Heidelberg, 2002.

[9] C. G. Cordero, S. Hauke, M. Mühlhäuser and M. Fischer, "Analyzing flow-based anomaly intrusion detection using Replicator Neural Networks," 2016. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/7906980.

[10] G. Kotani and Y. Sekiya, "Unsupervised Scanning Behavior Detection Based on Distribution of Network Traffic Features Using Robust Autoencoders," 2018. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8637559/figures#figures.

[11] P. Li, Z. Chen, L. T. Yang, J. Gao, Q. Zhang and M. J. Deen, "An Improved Stacked Auto-Encoder for Network Traffic Flow Classification," 2018. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8553650/metrics#metrics.

[12] P. Berezinski, B. Jasiul and M. Szpyrka, "An Entropy-Based Network Anomaly Detection Method," *Entropy,* 2015.

[13] C. G. Cordero, E. Vasilomanolakis, N. Milanov, C. Koch and D. Hausheer, "ID2T: A DIY dataset creation toolkit for Intrusion Detection Systems," 2015. [Online]. Available: https://ieeexplore.ieee.org/document/7346912.

[14] O. I. repository, "ID2T Github repo," [Online]. Available: https://github.com/tklab-tud/ID2T.

[15] "MAWI Working Group Traffic Archive," [Online]. Available: http://mawi.wide.ad.jp/mawi/.

[16] T. Kipf, "Graph Convolutional Networks," 2016. [Online]. Available: https://tkipf.github.io/graph-convolutional-networks/.

[17] T. N. Kipf and M. Welling, "Semi-Supervised Classification with Graph Convolutional Networks," 2016. [Online]. Available: https://arxiv.org/abs/1609.02907.

[18] M. Weber, J. Chen, T. Suzumura, A. Pareja, T. Ma, H. Kanezashi, T. Kaler, C. E. Leiserson and T. B. Schardl, "Scalable Graph Learning for Anti-Money Laundering: A First Look," 2018. [Online]. Available: https://arxiv.org/pdf/1812.00076.pdf.

[19] J. Chen, T. Ma and C. Xiao, "FastGCN: Fast learning with graph convolutional networks via important sampling," in *ICLR*, 2018.

[20] J. Zhou, G. Cui, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li and M. Sun, "Graph Neural Networks: A Review of Methods and Applications," 2019. [Online]. Available: https://arxiv.org/pdf/1812.08434.pdf.

[21] "MAWILab," [Online]. Available: http://www.fukuda-lab.org/mawilab/.

[22] R. Fontugne, P. Borgnat, P. Abry and K. Fukuda, "MAWILab: Combining Diverse Anomaly Detectors for Automated Anomaly Labeling and Performance Benchmarking," in *ACM CoNEXT '10*, Philadelphia, PA, 2010.

[23] V. Gadepally, J. Kepner, L. Milechin, W. Arcand, D. Bestor, B. Bergeron, C. Byun, M. Hubbell, M. Houle, M. Jones, P. Michaleas, J. Mullen, A. Prout, A. Rosa, C. Yee, S. Samsi and A. Reuther, "Hyperscaling Internet Graph Analysis with D4M on the MIT SuperCloud," *IEEE High Performance extreme Computing Conference (HPEC),* pp. 1-6, September 2018.

[24] A. Reuther, J. Kepner, C. Byun, S. Samsi, W. Arcand, D. Bestor, B. Bergeron, V. Gadepally, M. Houle, M. Hubbell, M. Jones, A. Klein, L. Milechin, J. Mullen, A. Prout, A. Rosa, C. Yee and P. Michaleas, "Interactive Supercomputing on 40,000 Cores for Machine Learning and Data Analysis," *IEEE High Performance extreme Computing Conference (HPEC),* pp. 1-6, September 2018.

[25] "yafscii(1) - Linux man page," [Online]. Available: https://linux.die.net/man/1/yafscii.

[26] N. Brownlee, C. Mills and G. Ruth, "Traffic Flow Measurement: Architecture," October 1999. [Online]. Available: https://tools.ietf.org/html/rfc2722.

[27] C. M. Inacio and B. Trammell, "YAF: Yet Another Flowmeter," in *Proceedings of LISA10: 24th Large Installation System Administration Conference*, 2010.

[28] B. Claise, B. Trammell and P. Aitken, "Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information," September 2013. [Online]. Available: https://tools.ietf.org/html/rfc7011.

[29] "MIT SuperCloud," [Online]. Available: https://supercloud.mit.edu/.

[30] M. Lopez-Martin, B. Carro, A. Sanchez-Esguevillas and J. Lloret, "Network Traffic Classifier With Convolutional and Recurrent Neural Networks for Internet of Things," *IEEE,* 2017.