Machine Learning in Anomaly-Based Network Intrusion Detection

Max Capote

Abertay University

School of Design and Informatics

MSc Ethical Hacking & Cyber Security

Abstract

Modern intrusion detection systems (IDS) are required to keep pace with increasingly sophisticated attacks. Signature-based systems alone are insufficient for detecting new methods of attack and anomaly-based systems typically suffer from high false-positive rates. In additional to this, traditional networks are not the only concern for security as the expansion of Internet-of-Things (IoT) in recent years has caused interconnected networks and the number of Internet-connected devices in general to explode in size. To begin strengthening defenses, an accurate IDS that can also detect previously unknown threats may be useful. This paper extends existing research in the application of machine learning for accurate anomaly-based network intrusion detection. The CICIDS2017 dataset is used as an aggregate for networks following a more traditional architecture of workstations, servers, switches, routers, *etc*. and the BoT-IoT dataset is used as an aggregate for network architectures that are influenced by IoT and may contain unconventional devices, such as mobile devices. An XGBoost model as well as a feed-forward neural network were constructed to perform binary and multiclass classification on each dataset. For CICIDS2017, XGBoost achieved the highest accuracy for both binary classification at 99.93% and multiclass classification at 99.92% For BoT-IoT, XGBoost and neural network achieved 99.99% accuracy on both types of classification. Additional metrics recorded in this paper include precision and recall scores in the interest of observing false-positive and false-negative rates, respectively. Overall, machine learning is a worthwhile technology to continue researching and adopting for intrusion detection in traditional and IoT-based networks.

Keywords: Internet of Things, Intrusion Detection, Anomaly-Based Detection, Machine Learning, Ensemble Learning, Neural Network

Table of Contents

List of Tables

1 INTRODUCTION

Technology has historically maintained a rapidly shifting landscape. Unfortunately, the speed of development comes with unintended consequences. Today, cyber security is very much a concern of corporations, governments, and individuals (Capgemini Research Institute, 2019). A cyber threat can be defined as "…a malicious act that seeks to damage data, steal data, or disrupt digital life in general" (Taylor, 2020). Some consequences following successful exploitation of technical vulnerabilities or personnel include equipment failure, disclosure of national security information, theft of valuable personal data, denial of services, and destruction of a network (Taylor, 2020).

Numerous attacks are launched every day, and the attack strategies of advanced persistent threats are becoming increasingly sophisticated in response to current defensive measures (Taylor, 2020). Even nonpersistent threats are incredibly harmful, and increasingly common. Ransomware, for instance, has exploded in popularity in recent years. In 2018, 39% of data breaches around the world caused by malware involved ransomware, accounting for more than eight billion dollars of extortion and damage (IRM, 2019).

To further complicate the situation, networks are expanding rapidly via technology categorized as Internet of Things (IoT) and techniques that have historically been effective for securing traditional networks cannot be applied so easily to networks heavily incorporating IoT devices (Hassan and Noor, 2018). The challenges mostly arise from the many different types of devices that are becoming interconnected, the sort of fragmentation that comes with the multitude of networking protocols, and the scale of an IoT network architecture in terms of nodes on the network (Hassan and Noor, 2018). Research in securing IoT is ongoing, but with the many potential vectors that exist (e.g. inadequate authentication, hardcoded credentials, and insecure data transmission), there is a need for innovative approaches that can service complex network architectures where traditional approaches fall short.

Given the consequences of increasingly intricate attacks as well as the insecurity of emerging technology, cyber security continues to grow in difficulty. In a 2019 cyber security report, Capgemini shares that 56% of participating organizations already claim that analysts are becoming overwhelmed given that the volume as well as a variety of data being encountered are of such high magnitude for effective monitoring (Capgemini Research Institute, 2019). As a result, 61% of participating organizations acknowledge that artificial intelligence (AI) will be necessary for the successful identification of critical threats in the future (Capgemini Research Institute, 2019). For 64% of these organizations that are already innovating, Capgemini reports that the implementation of AI lowers the cost of intrusion detection and response at an average of 12% (Capgemini Research Institute, 2019). Moreover, overall detection and response time has been reduced by up to 12%, which results nicely in higher efficiency for analysts as more time can be dedicated to analyzing attempted intrusions (Capgemini Research Institute, 2019). Being such a potentially impactful use case, forms of AI in intrusion detection have already received much attention from researchers. This paper aims to extend existing research by assessing the effectiveness of machine learning, a subset of AI, in traditional and IoT networks. It employs two recently published datasets from within the last three years and evaluates the potential viability of the technology in its current state for network defense.

2 BACKGROUND

2.1 INTRUSION DETECTION

An intrusion detection system (IDS) is essentially a software application that monitors events for signs of malicious activity. More specifically, the malicious activity includes attempts to compromise confidentiality, integrity, or availability, or bypass security mechanisms of a computer or network (Liao et al., 2012). Two of the most common forms are host-based intrusion detection systems (HIDS) and network-based intrusion detection systems (NIDS). The difference between the two is as follows. A host-based system is fixed to a specific host and can detect intrusions by monitoring files in the operating system. A network-based system sits between machines and monitors network traffic for possible intrusions (Liao et al., 2012).

As far as methods of detection, two of note are signature-based detection and anomaly-based detection. Signature-based detection is a simple and effective solution wherein patterns or strings of known threats are compared against captured data to determine intrusions (Liao et al., 2012). This being the case, signature-based detection is unable to detect unknown attacks, or in other words, attacks that do not yet have a signature available to the IDS, which creates a need for signatures to be kept up to date. Anomaly-based detection, on the other hand, is effective against unknown attacks as the IDS, in this case, assesses activity for intrusions based on profiles compiled of the known or expected behavior (Liao et al., 2012). Unknown behavior is anomalous, and therefore, considered intrusion. The longstanding issue of this methodology has been inaccurate detection because of high false-positive rates (identifying benign activity as malicious) and false-negative rates (failing to identify malicious activity) (Liao et al., 2012). Researchers are seeking to improve the detection of this kind by using machine learning to create models that correctly identify all activity as often as possible.

2.2 MACHINE LEARNING

Machine learning is a subset of AI that is centered on the idea that systems can learn from data or experience and successfully make decisions without explicitly being programmed to do so (SAS Institute, 2020). Today, machine learning is of interest to researchers and professionals in various fields (e.g. cyber security, financial services, and health care) as conditions are at last favorable for deploying the technology. That is, large volumes of data are available, computational processing is cheaper and more powerful, and data storage is affordable (SAS Institute, 2020). Machine learning can potentially be of great use when confronted with complex problems that do not have sufficient traditional approaches (e.g. anomaly-based intrusion detection) and when deployed in an environment that may change unpredictably as systems have the ability to adapt to new data. That being said, machine learning can be difficult to effectively use if the data needed to train a model is of poor quality (e.g. insufficient quantity or many missing values) (Géron, 2019).

Some broad categories are used to categorize different types of machine learning models. The first, supervised learning, involves feeding an algorithm the given training data which contains the desired solutions, or labels (Géron, 2019). It is explicitly telling the algorithm what correct solutions are. A common task for supervised learning is classification (Géron, 2019). An algorithm may distinguish between two classes, which is binary classification, or between more than two classes, which is multiclass classification (Géron, 2019). The second category, unsupervised learning, involves feeding an algorithm the given training data and allowing the algorithm to determine the appropriate labels (Géron, 2019). Rather than telling the algorithm what correct solutions are, it will determine the solutions on its own based on the connections it finds. The third category, semi-supervised learning, involves handling training data that is only partially labeled (Géron, 2019). Many algorithms that resort to semi-supervised techniques combine elements of supervised and unsupervised algorithms (Géron, 2019). Lastly, reinforcement learning involves an agent, rather than a model, that gradually learns which actions given an environment will yield the best rewards. This is accomplished through trial and error, and the agent is considered to have learned the optimal policy, which is the set actions that maximize the rewards by the end (SAS Institute, 2020).

2.3 Dataset Overview

2.3.1 CICIDS2017

The CICIDS2017 dataset was created by the Canadian Institute for Cybersecurity at the University of New Brunswick in Canada in 2017 (Ghorbani et al., 2018). The motivation for generating this dataset came from a desire to address issues of earlier datasets as well as to provide a comprehensive dataset that is publicly available. Primary issues of previous datasets include outdated traffic, lack of attack diversity, anonymized data, and unrealistic traffic in some cases (Ghorbani et al., 2018). Especially given the frequency at which malware and attack strategies adapt to defenses, periodically updating a dataset for such a purpose as intrusion detection is paramount. KDD99 and NSL-KDD, for example, are two very commonly used datasets in machine learning, but they are more than twenty and ten years old, respectively, whereas CICIDS2017 is three years old at the time of this paper (Ghorbani et al., 2018).

The simulated environment combines two networks, one solely comprising of the victims and the other solely comprising of the attackers. The victim network is complete with ten workstations as well as a few servers, one firewall, one router, and two switches (Ghorbani et al., 2018). Operating systems in use here include various versions of Windows, Macintosh, and Linux. The attack network, in contrast, includes one router and switch for four workstations, three of which running Windows 8 and the fourth running a version of Kali Linux (Ghorbani et al., 2018). Data generation began 09:00 on Monday (3 July 2017) and ran continuously for five days until 17:00 on Friday (7 July 2017), with attacks executing throughout the period. Attacks included brute-force, denial-of-service (DoS) and distributed denial-of-service (DDoS), infiltration attacks, botnet attacks, and various web attacks, including cross-site scripting (XSS) and SQL injection (SQLi) (Ghorbani et al., 2018). Of note, the dataset is completely labelled and represents multiclass data.

2.3.2 BoT-IoT

The BoT-IoT dataset was created by a team in the Research Cyber Range lab of the University of New South Wales in Canberra, Australia (Koroniotis et al., 2018). The motivation for generating this dataset came from a desire to provide a publicly available dataset for researching formidable security measures in IoT networks given that existing solutions are insufficient. In contrast to previously published datasets, this work offers recent and complex attack diversity as well as realistic network traffic featuring botnet activity (Koroniotis et al., 2018). A noted limitation, however, is that the use of a virtualized environment left the team unable to launch thorough attacks against firmware and hardware of victim machines (Koroniotis et al., 2018).

The test environment contains victim and attacker virtual machines with a couple of firewalls, a switch, and a network tap. The victim network has two servers, one running a version of Ubuntu and the other running Metasploitable (Koroniotis et al., 2018). In addition to these, machines are running a mobile version of Ubuntu, Windows 7, and a version of Ubuntu to serve as the network tap (Koroniotis et al., 2018). Normal traffic was generated with the Ostinato packet generation tool and Node-RED was used for simulating IoT network behavior. Using JavaScript code, behavior was mimicked for IoT sensors capable of reporting such information as temperature, pressure, and humidity (Koroniotis et al., 2018). The attacker network has four machines running a version of Kali Linux. Attacks included DoS and DDoS over UDP, TCP, and HTTP, port and service scans, OS fingerprinting, keylogging, and data exfiltration (Koroniotis et al., 2018). This dataset is also completely labelled and represents multiclass data.

2.4 Algorithm Overview

XGBoost and neural network are the two algorithms chosen for this research, and what follows is a brief description of each. The justification for selecting these two algorithms is described in detail in the next section, literature review.

2.4.1 XGBoost

XGBoost, or Extreme Gradient Boosting, is an ensemble algorithm that uses a gradient boosting framework with decision trees for weak learners (Morde, 2019). Boosting methods generally train weak learners sequentially, each attempting to correct the previous one. Gradient boosting specifically fits each new weak learner on the residual errors made by the previous one (Géron, 2019). XGBoost differs from the traditional gradient boosting framework by offering system optimizations and algorithmic improvements. For instance, when training a more complex model, it is put through both Lasso and Ridge regularization to avoid overfitting (Morde, 2019). Interestingly, too, the traditional stopping criterion for tree splitting depends on the negative loss criterion at the point of the split. XGBoost, on the other hand, uses a hyperparameter called ‘max_depth’ instead of addressing criterion first, which results in a depth-first approach that prunes trees backwards and improves computational performance (Morde, 2019).

2.4.2 Neural Network

A neural network, or neural net, is an algorithm designed to emulate the way the human brain functions (Bermudez, 2017). In a human brain, some neurons are connected by synapses. When thinking occurs, inputs travelling across synapses from one neuron to the next cause those neurons to fire. At a high level, when modeling this in source code, neural networks similarly contain input neurons and output neurons that are connected by synapses (Bermudez, 2017). The synapses, in this case, bear weights that affect forward propagation. Learning for a neural network occurs when these weights are updated during backward propagation (Bermudez, 2017). Because this learning process is conducted on each piece of data in a training set, a neural network benefits from large quantities of data as well as datasets that are varied (Bermudez, 2017). The more a neural network can learn, the better it will be at predicting outputs.

2.5 LITERATURE REVIEW

The main decisions for this project, namely which algorithms to experiment with and what type of classification to perform, were inspired by previous works that made use of at least one of the same datasets, CICIDS2017 or BoT-IoT.

A thesis paper by Aksu et al. (2018) from Istanbul University focused solely on binary classification to differentiate between benign network traffic and network traffic indicative of a distributed denial-of-service (DDoS) attack (Aksu et al., 2018). The approach used three traditional classifiers, which were support vector machine (SVM), k-nearest neighbors (KNN), and decision tree (DT) (Aksu et al., 2018). The future work noted utilizing the entire dataset rather than subsets as well as devising a deep learning algorithm (Aksu et al., 2018).

Another paper by research group D'hooge et al. (2019) from Ghent University also focused on binary classification (D'hooge et al., 2019). This is reasonable but calls attention to a possible gap in research since Aksu et al. (2018) also did not conduct multiclass classification. That being said, the project by D'hooge et al. (2019) did make use of the entire dataset and expanded the classification problem to benign network traffic versus network traffic indicative of an attack, of which there are many different types (D'hooge et al., 2019). This approach notably made use of nine diverse classifiers, including DT, random forest (RF), bagging, Adaboost, KNN, n-centroid, LinearSVC, RBFSVC, and logistic regression, but there was lack of deep learning (D'hooge et al., 2019). To add, tree-based classifiers reportedly performed the best, which included the ensemble methods (D'hooge et al., 2019). Hence, introducing a different boosting algorithm, XGBoost, and providing findings from multiclass classification can cover the gap in their research.

A paper by Dogdu and Faker, a pair of researchers from Cankaya University, addresses the noted concerns up to this point in the research (Dogdu and Faker, 2019). Both binary and multiclass classification tasks are carried out with a deep neural network (DNN) and additional ensemble methods in the form of RF and gradient boosted trees (GBT) (Dogdu and Faker, 2019). The deep neural network showed promise with high accuracy in binary classification and the highest accuracy in multiclass classification. While experimenting with multiclass classification, though, all normal traffic was removed from the dataset since this traffic accounts for a large amount of the data (Dogdu and Faker, 2019). With that, multiclass classification using all available labels remains a lesser explored endeavor, and in this paper, normal traffic is included when using either dataset despite its share in CICIDS2017. What is more, the GBT model did not support multiclass classification at the time of the experiment in the chosen computing environment, Apache Spark (Dogdu and Faker, 2019). This further justifies the use of XGBoost as it is regarded as a more efficient algorithm than traditional gradient boosting and the classifier in this paper will be tasked with multiclass classification.

This research is based partially on the research done by Al-Nemrat et al. (2019) who submitted work to the Institute of Electrical and Electronics Engineers (IEEE) proposing an interesting and convincing hybrid framework for intrusion detection. The framework can be conceptualized into two parts. The focus, though, was the DNN module (Al-Nemrat et al., 2019). Although many datasets were used in this experiment, CICIDS2017 was one, which addresses a concern for lack of deep learning application. In their testing, the DNN overall performed well compared to tested traditional classifiers, which included DT, Adaboost, RF, logistic regression, Naïve Bayes, KNN, and SVM (Al-Nemrat et al., 2019). Also, worth noting, both binary and multiclass classification were performed.

Considering the data first, this team decided to scale down the dataset tremendously for training and testing with only 93500 training instances and 28481 testing instances (Al-Nemrat et al., 2019). In the breakdown of the attack types for each set, the team have removed infiltration attacks (thirty-six instances) entirely unless those instances were reclassified as part of another attack type, but this is unspecified (Al-Nemrat et al., 2019). To reiterate, there is value to experimenting with the full dataset, or at least as much as possible after necessary preprocessing.

Looking at the DNN, like Dogdu and Faker, it followed a feed-forward approach with each hidden layer fully connected to the next layer (Al-Nemrat et al., 2019). The overall architectures, though, varied quite a bit. For Al-Nemrat et al. (2019), the lowest number of hidden layers tested was one and the highest number was five (three layers were found to be optimal). All told and in hierarchical order, the largest network contained 1024 neurons in the first hidden layer, 768 in the second, followed by 512, 256, and finally 128 before the output layer (Al-Nemrat et al., 2019). Layers for batch normalization and dropout were also included between each hidden layer for speed and to avoid overfitting (Al-Nemrat et al., 2019). In sharp contrast, the hidden layers in the model proposed by Dogdu and Faker contained, in hierarchical order, 128 neurons, followed by 64 neurons, and finally 32 neurons before reaching the output layer (Dogdu and Faker, 2019). There is no mention of batch normalization nor dropout (Dogdu and Faker, 2019). These are two very different approaches, so despite the extensive work conducted by Al-Nemrat et al. (2019), utilizing more of the data available as well as constructing a neural network is worth the effort to explore how successful this approach could be if given a wealth of network traffic for learning and to provide what could be a third valid neural network model for CICIDS2017.

For the BoT-IoT dataset, deep learning was a more deeply explored approach for research. For the 2019 IEEE 24th Pacific Rim International Symposium on Dependable Computing (PRDC), for instance, Baig et al. (2019) prepared a paper that proposed a feed-forward deep neural network (FFDNN) for binary and multiclass classification which delivered high accuracy (Baig et al., 2019). This team made use of the available PCAP files instead of CSV and opted to extract 2% of the data used (Baig et al., 2019). One million packets per subcategory of an attack were used if there were over one million available, otherwise all packets for a subcategory were used. After preprocessing, 7310546 instances of data remained, which is slightly under two times as large as the collection of instances used in this paper (Baig et al., 2019). From the available CSV files, Koroniotis et al. (2018) provide a set of files which encompasses 5% of the total dataset, which is very reasonable to work with as far as space and number of instances (Koroniotis et al., 2018). Ferraga et al. (2019) also used the provided 5% of the total dataset and were quite comprehensive with deep learning approaches (Ferraga et al., 2019). The FFDNN proposed by Baig et al. (2019) was one approach, containing input and output layers as well as two hidden layers with 512 neurons each, and Ferraga et al. (2019) offer further experimentation by applying several more deep learning approaches, including a recurrent model, restricted Boltzmann machine, deep Boltzmann, deep belief, convolutional, and deep autoencoders (Baig et al., 2019; Ferraga et al., 2019). Baig et al. (2019) noted that the complexity of their model was unable to capture the variation of different classes while training or began overfitting since variations of their architecture did not increase classification accuracy (Baig et al., 2019). As far as overall results from Ferraga et al. (2019), the convolutional neural network had the overall highest detection rate and the lowest false alarm rate for a supervised approach while deep autoencoders had the overall highest detection rate and lowest false alarm rate for an unsupervised approach (Ferraga et al., 2019). In short, there has been much investigation into deep learning with BoT-IoT. The extension of current research may be experimenting with a strong ensemble approach like XGBoost to see to what degree other algorithms may be viable for intrusion detection in IoT given this dataset.

A final paper worth considering written by Mahmoud and Ullah and submitted to Ontario Tech University proposed an interesting two-level system for recognizing anomalous activity in IoT networks using the BoT-IoT dataset for experimentation (Mahmoud and Ullah, 2020). In this approach, no deep learning took place. The first level was made to differentiate normal network traffic from attack and made use of a DT classifier. The second level, now using RF, was made for further classifying the category or subcategory of a detected anomaly. The accuracy, precision, recall, and F scores for this experiment were very promising with 99.99% across metrics for the first level and 99.90% across metrics for the second level (Mahmoud and Ullah, 2020). The use of a tree-based classifier and tree-based ensemble worked quite well, but the team noted that in the future they would attempt to improve the accuracy of insignificant classes, which is where XGBoost may be able to extend research while tasked with multiclass classification. This paper is not proposing a similar two-level design, just reporting what kind of usefulness may be extracted from additional algorithms.

3 METHODOLOGY

The high-level objectives for this procedure are as follows:

- Establish the environment used for experimentation
- Sufficiently clean and process both datasets being used
- Construct, train, and fine-tune machine learning models using the chosen algorithms

3.1 LAB SETUP

Experimentation was carried out on a single laptop computer with the software and hardware specifications in Table 1. To summarize, the computer was running a 64-bit version of the Windows 10 operating system and used the Visual Studio Code text editor alongside Python to handle data cleaning and processing, model construction, and model evaluation (Table 1).

Table 1: Lab environment specifications.

Operating System	Windows 10
System Type	x64-based
Processor	Intel Core i7-6500U (2 cores and 4 logical processors)
Processor Speed	2.50 GHz
Installed Physical Memory (RAM)	16.0 GB
Source Code Editor	Visual Studio Code 1.45.1
Programming Language	Python 3.8.3

3.2 DATA CLEANING AND PREPROCESSING

Preprocessing was started with CICIDS2017. The general details of the dataset are provided in Table 2.

Table 2: General overview for the CICIDS2017 dataset.

Name	CICIDS2017
Type	Multiclass
Year Published	2017

Number of Records	2830743
Number of Features	79
Number of Classes	15

The dataset is received in eight separate files. It is combined into a single file for ease of use (Table 3). The large volume of data, of course, still requires a fair bit of overhead for loading and processing. The details on the individual files are provided in Table 3 and highlight which day of the week each attack type occurred.

Table 3: Key details of individual files provided for the CICIDS2017 dataset.

Filename	**Day of Week**	**Number of Records**	**Traffic Encountered**
Monday-WorkingHours.pcap_ISCX.csv	Monday	529918	Benign
Tuesday-WorkingHours.pcap_ISCX.csv	Tuesday	445909	Benign, FTP-Patator, SSH-Patator
Wednesday-workingHours.pcap_ISCX.csv	Wednesday	692703	Benign, DoS Hulk, DoS GoldenEye, DoS slowloris, DoS Slowhttptest, Heartbleed
Thursday-WorkingHours-Morning-WebAttacks.pcap_ISCX.csv	Thursday	170366	Benign, Web Attack – Brute Force, Web Attack – XSS, Web Attack – Sql Injection
Thursday-WorkingHours-Afternoon-Infilteration.pcap_ISCX.csv	Thursday	288602	Benign, Infiltration
Friday-WorkingHours-Morning.pcap_ISCX.csv	Friday	191033	Benign, Bot
Friday-WorkingHours-Afternoon-PortScan.pcap_ISCX.csv	Friday	286467	Benign, PortScan

Friday-WorkingHours-Afternoon-DDos.pcap_ISCX.csv	Friday	225745	Benign, DDoS

There is great class imbalance among the fifteen classes, as shown in Table 4. Most notably, benign traffic accounts for 2273097 instances out of 2830743 total instances (Table 4). There are also classes with comparatively low representation. Heartbleed attacks, for instance, only account for eleven instances in the dataset (Table 4).

Table 4: Class imbalance of the CICIDS2017 dataset.

Class Label	**Number of Records**
BENIGN	2273097
DoS Hulk	231073
PortScan	158930
DDoS	128027
DoS GoldenEye	10293
FTP-Patator	7938
SSH-Patator	5897
DoS slowloris	5796
DoS Slowhttptest	5499
Bot	1966
Web Attack – Brute Force	1507
Web Attack – XSS	652
Infiltration	36
Web Attack – Sql Injection	21
Heartbleed	11

The recommendation to reassign labels based on an analysis of CICIDS2017 from Borah and Panigrahi is followed to address the class imbalance (Borah and Panigrahi, 2018). This step involved merging classes that have similar characteristics and behavior. An example is grouping all denial-of-service-based attacks by reassigning these instances the label of ‘DoS/DDoS’ (Table 5). The complete reassignment is provided in Table 5.

Table 5: Relabeling to address class imbalance in the CICIDS2017 dataset.

New Label	**Old Label**	**Number of Instances**
Normal	BENIGN	2271320
DoS/DDoS	DDoS, DoS Hulk, DoS GoldenEye, DoS slowloris, DoS Slowhttptest, Heartbleed	379748
Port Scan	PortScan	158804
Brute-Force	FTP-Patator, SSH-Patator	13832
Web Attack	Web Attack – Brute Force, Web Attack – XSS, Web Attack Sql Injection	2180
Botnet ARES	Bot	1956
Infiltration	Infiltration	36

This dataset contains infinity as well as missing values, but these luckily account for a negligible number of instances (2867). To correct this, the infinity values were replaced with NaN first, then all instances containing NaN were dropped, which was approximately 0.101% of the dataset. Features with no variance were removed too, of which there were eight: ‘ Bwd PSH Flags,’ ‘ Bwd URG Flags,’ ‘Fwd Avg Bytes/Bulk,’ ‘ Fwd Avg Packets/Bulk,’ ‘ Fwd Avg Bulk Rate,’ ‘ Bwd Avg Bytes/Bulk,’ ‘ Bwd Avg Packets/Bulk,’ and ‘Bwd Avg Bulk Rate.’ Just to clarify, there are some features that begin with a whitespace character as part of the feature name, and as this does not make a functional difference in classification, feature names are used as they are given. As a note, the updated class labels are used for multiclass classification, and binary classification requires a separate preprocessing step to reassign all attack traffic the label of ‘Attack.’ On the note of separate preprocessing steps, the neural network accepts only numerical input, so the categorical target labels are binarized before use with this model. Additionally, the features are scaled beforehand, but are not when using XGBoost. In either case, training and testing sets are created using an 80/20 split of the data, respectively.

Looking at the BoT-IoT dataset, general details are provided in Table 6. The used version represents 5% of the complete dataset.

Table 6: General overview for the BoT-IoT dataset.

Name	**BoT-IoT**
Type	Multiclass
Year Published	2018
Number of Records	3668522
Number of Features	46
Number of Classes	5

Like CICIDS2017, this dataset is received in multiple files. The files are combined into a single file for clarity (Table 7). The individual files are detailed in Table 7.

Table 7: Key details of the individual files provided for the BoT-IoT dataset.

Filename	**Number of Records**	**Traffic Encountered**
UNSW_2018_IoT_Botnet_Full5pc_1.csv	1000000	DoS
UNSW_2018_IoT_Botnet_Full5pc_2.csv	1000000	DoS, DDoS
UNSW_2018_IoT_Botnet_Full5pc_3.csv	1000000	DDoS
UNSW_2018_IoT_Botnet_Full5pc_4.csv	668522	DDoS, Reconnaissance, Normal, Theft

There is more data to process than in CICIDS2017, even at only 5% of the original dataset, but far fewer class labels. Unlike CICIDS2017, normal traffic accounts for very little network activity in this dataset as DDoS and DoS attack traffic account for the major share of instances (Table 8). The class imbalance is further detailed in Table 8.

Table 8: Class imbalance of the BoT-IoT dataset.

Class Label	**Number of Records**
DDoS	1926624
DoS	1650260
Reconnaissance	91082

Normal	477
Theft	79

Although classification in this paper will make use of the ‘category’ feature, it may be worth mentioning the ‘subcategory’ feature, which further specifies the type of attack being conducted. Theft, for example, can be further examined as attempted data exfiltration or keylogging (Table 9).

Table 9: Overview of the subcategories for traffic encountered in the BoT-IoT dataset.

‘category’	**‘subcategory’**	**Number of Records**
DoS/DDoS	UDP	1981230
	TCP	1593180
	HTTP	2474
Reconnaissance	Service_Scan	73168
	OS_Fingerprint	17914
Normal	Normal	477
Theft	Keylogging	73
	Data_Exfiltration	6

At the very least, there are no infinity nor missing values to clean. There are two features, however, that contain mixed data types; ‘sport’ and ‘dport.’ These features contain values that are either string or int, and some string values are hexadecimal values (e.g. 10, ‘10’, and ‘0x10’). For this issue, the strings of hexadecimal values are converted to ints of decimal values and then all remaining strings are converted to ints too. Apart from this, the dataset also contains categorical features, so one-hot encoding is used to ensure the data is appropriate for a model. For the labels, in being consistent with CICIDS2017, the traffic for the labels ‘DoS’ and ‘DDoS’ are combined into one label, ‘DoS/DDoS,’ since these attacks share common behavior (Table 10). The updated class labels as well as number of instances per class are provided in Table 10.

Table 10: Relabeling to group together attacks of similar behavior in the BoT-IoT dataset.

New Label	**Old Label**	**Number of Instances**
Normal	Normal	477

DoS/DDoS	DDoS, Dos	3576884
Reconnaissance	Reconnaissance	91082
Theft	Theft	79

The updated class labels are used for multiclass classification, and binary classification once again requires a separate preprocessing step to reassign all attack traffic the label of ‘Attack.’ Lastly, training and testing sets are again created using an 80/20 split of the data, respectively, and the neural network model requires the additional binarizing of target labels and feature scaling to be able to use the data effectively.

3.3 Model Construction

To mention again, Python is used to write the programs for this project. This is because of personal familiarity with the language and there is much support available for machine learning in Python with such libraries as Scikit-learn and TensorFlow.

To fine-tune each model, 4-fold cross-validation is used with help from the KFold function available from Scikit-learn. As the source code shows, for each fold, a new model is created and trained on a portion of the data sampled, then evaluated on the remaining data from the sample. The purpose of cross-validation is essentially to have a better estimate of a model’s performance against unseen data as opposed to training on all training data and then moving immediately to evaluation on the testing data. Given the stochastic nature of training, results can fluctuate, making them unreliable while still fine-tuning. During cross-validation, however, a model is subject to multiple iterations of training and evaluation and never touches the testing data, resulting in more reliable metrics to observe while fine-tuning.

Configuration details of each model for each dataset are provided in Appendix A. To give some summary information, working with CICIDS2017 was a straight-forward process. Most notably, the XGBoost model uses 125 base estimators with a learning rate of 0.3. The neural network has an input layer for 70 features which feeds into three fully connected hidden layers and a final output layer. The hidden layers have a hierarchical structure of 256 followed by 128 and then 64, with each utilizing ReLU for an activation function. The output layer contains 1 neuron using sigmoid activation function for binary classification and 7 neurons using softmax activation function for multiclass classification.

BoT-IoT was more difficult to construct models for because nearly perfect to perfect scores (0.9999 to 1.0) were encountered across metrics with different configurations, which is uncommon. Although such results were achieved in related research, the models were possibly overfitting the training data. The data provided may be quite simple, and both XGBoost and neural networks are generally quite complex approaches to machine learning problems. To combat these issues, the complexity of each model was simplified as much as possible and additional randomness was introduced. For XGBoost, the model contains only 10 base estimators (default value is 100) and the maximum depth of a tree is only set to 1. This model was also made to be more conservative by tuning ‘min_child_weight’ and ‘gamma,’ both of which are set to 20. Lastly, the added randomness is achieved by tuning ‘subsample’ and ‘colsample_bytree,’ both of which are set to 0.5. For the neural network, the input layer accommodates 177 features, a number the result of one-hot encoding, followed by a single hidden layer containing 16 neurons with each neuron utilizing ReLU. To further address overfitting, the hidden layer is followed by a dropout layer which has a dropout rate of 0.5 for a simple form of regularization. The output layer contains 1 neuron using sigmoid for binary classification and 4 neurons using softmax for multiclass classification.

4 RESULTS

To assess the success of the constructed models, accuracy, precision, and recall are employed for metrics. Accuracy is very simply the total number of correct predictions divided by the total number of predictions made. Although this is a commonly used metric, a reasonable intrusion detection model in this context is one that is capable of maximizing precision (minimizing number of false-positives) and maximizing recall (minimizing number of false-negatives) as much as possible (Brownlee, 2020). To add, accuracy alone is misleading when classifying imbalanced data because considerably high accuracy may be achieved if the majority class comprises a large portion of the data (Brownlee, 2020). In other words, the classifier may learn the majority class very well, but fail to learn the minority class as intended. In binary classification, precision is calculated as the number of true-positives divided by the total number of true-positives and false-positives:

Precision = True-Positives / (True-Positives + False-Positives)

In multiclass classification, the calculation expands to include results across classes, which becomes the sum of true-positives across all classes divided by the sum of true-positives and false-positives across all classes:

Precision = Sum True-Positives All Classes / (Sum True-Positives + False-Positives All Classes)

Recall in binary classification is calculated as the number of true-positives divided by the total number of true-positives and false-negatives:

Recall = True-Positives / (True-Positives + False-Negatives)

Recall expands to multiclass classification in much the same fashion as precision, which means taking the sum across classes as needed:

Recall = Sum True-Positives All Classes / (Sum True-Positives + False-Negatives All Classes)

4.1 BINARY CLASSIFICATION

XGBoost notably performed best against CICIDS2017, and the results for BoT-IoT are comparable between the two models (Table 11).

Table 11: Results of binary classification on both datasets.

Model	Dataset	Accuracy	Precision	Recall
XGBoost	CICIDS2017	99.93%	99.93%	99.93%
	BoT-IoT	99.99%	99.99%	99.99%

Neural Network	CICIDS2017	99.54%	99.75%	99.68%
	BoT-IoT	99.99%	99.99%	99.99%

4.2 Multiclass Classification

XGBoost again notably performed best against CICIDS2017, and the results for BoT-IoT are once more comparable between the two models (Table 12).

Table 12: Results of multiclass classification on both datasets.

Model	Dataset	Accuracy	Precision	Recall
XGBoost	CICIDS2017	99.92%	99.92%	99.92%
	BoT-IoT	99.99%	99.99%	99.99%
Neural Network	CICIDS2017	99.46%	99.46%	99.46%
	BoT-IoT	99.99%	99.99%	99.99%

5 DISCUSSION

XGBoost and neural network scored above 99% across metrics for binary and multiclass classification on both datasets. Focusing first on CICIDS2017 research, extending Aksu et al. (2018) and D'hooge et al. (2019), deep learning and a strong ensemble method are used effectively on two types of classification. When compared to the model by Al-Nemrat et al. (2019), in binary classification, highest scores were achieved by networks using one layer, scoring 96.3% accuracy, one layer, scoring 90.8% precision, and three layers, scoring 97.9% recall (Al-Nemrat et al., 2019). In comparison, the constructed neural network by this research achieved 99.54% accuracy, 99.75% precision, and 99.68% recall. In multiclass classification, the network using three layers was the best across metrics with 96.2% accuracy, 97.2% precision, and 96.2% recall (Al-Nemrat et al., 2019). The neural network by this research, however, ended with 99.46% across metrics for multiclass classification.

Looking at Dogdu and Faker, the neural network constructed in this project provides improved accuracy (97.71% versus 99.54%) as well as high precision and recall scores for binary classification using the full dataset (no feature selection) (Dogdu and Faker, 2019). As far as multiclass classification, using the full dataset, Dogdu and Faker achieved 99.56% accuracy with their constructed model versus the neural network by this research scoring 99.46% accuracy (Dogdu and Faker, 2019). The difference could possibly be the decision to include the normal network traffic data in this paper as this was the majority class by a large margin. Moreover, XGBoost achieved an accuracy of 99.93% for binary classification versus the GBT classifier from Dogdu and Faker which scored 99.81%, again using the full dataset (Dogdu and Faker, 2019). With feature selection, though, Dogdu and Faker achieved 99.99% accuracy using GBT (Dogdu and Faker, 2019). Furthermore, XGBoost proved to be a useful GBT ensemble in multiclass classification by scoring 99.92% across metrics.

In regards to BoT-IoT, through much research in the application of deep learning by Baig et al. (2019) and Ferraga et al. (2019), XGBoost provides solid evidence that an ensemble method can also achieve high performance using this dataset while performing binary or multiclass classification. The ensemble used by Mahmoud and Ullah, RF, likewise achieved high accuracy, but scored slightly worse with 99.90% compared to XGBoost scoring 99.99% (Mahmoud and Ullah, 2020). Even given the challenges presented by this dataset, though, if it is truly representative of botnet-related and other malicious traffic on IoT-based networks, the application of comparatively weaker algorithms may warrant further research as both the XGBoost and neural network models had to be drastically reduced in complexity to lessen the possibility of overfitting. If simpler algorithms can perform nearly or as strongly as either one employed in this paper, machine learning could be very impactful in securing IoT-based networks.

On that note, assessing machine learning for network defense, this technology is viable for use even in its current state looking at the great results achieved in this paper, and because of how such strong performances are consistent across research. Machine learning is clearly imperfect as models do generally exhibit some margin of error, but machine learning can still be significant in certain situations. In IoT-based networks, for instance, the data gathered and reviewed suggests that machine learning would be an incredibly accurate method of intrusion detection with minimal false-positives and false-negatives.

Unfortunately, as promising as machine learning is, development can be troublesome, for lack of a better word. Since data is monumental for machine learning, a reasonable set of data needs to be procured. If an appropriate dataset is published, that would be convenient, but if not, a dataset may need to be created, which would require additional time and resources. What is more, after data has been cleaned and processed effectively for a model, the process of tuning a model can be frustrating as the feedback the programmer receives after adjusting hyperparameters may come very slowly. Especially when dealing with a complex model and a large dataset with millions of instances, a single fold in k-fold cross-validation may require hours to complete. Nonetheless, machine learning stands to be quite useful for intrusion detection in different types of networks so the time required for creating a model may be a worthwhile trade-off for the exceptional accuracy and infrequent false classifications.

6 Conclusion

In closing, different networks today need to be able to detect intrusions accurately, and the solutions currently available have unfortunate disadvantages. Machine learning presents an innovative approach to anomaly-based intrusion detection that can achieve great accuracy with minimal false-positives and false-negatives. From this paper, metrics above 99% were achieved while performing binary and multiclass classification using the CICIDS2017 dataset and nearly perfect metrics were achieved (99.99%) when using the BoT-IoT dataset. Performing the classification tasks were an XGBoost model as well as a feed-forward neural network. Finally, further research in machine learning will be advantageous in the future, but the technnology is acceptable for adoption in its current state too for network intrusion detection.

6.1 Future Work

Processing the datasets further by employing a feature selection algorithm is a subject of future work as this is a significant component for successful machine learning models. To extend this paper to some degree, incorporating more datasets, if appropriate ones are published and become accessible, is a possibility. Using different neural network architectures, like recurrent or convolutional, is also worth considering. If deep learning becomes the primary interest over other methods, then semi-supervised or unsupervised experiments may be considered (e.g. generative adversarial networks). Lastly, it would be interesting to deploy an application using a model trained on one of the datasets used in this paper in a test environment and evaluate its detection ability in real-time either with automated or manual attacks.

REFERENCES

Al-Nemrat, A., Alazab, M., Poornachandran, P., Soman, K. P., Venkatraman, S., and Vinayakumar, R. (2019). *Deep Learning Approach for Intelligent Intrusion Detection System*. [online] Available from: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8681044 [Accessed 30 June 2020].

Aksu, D., Atmaca, T., Aydin, M., and Üstebay, S. (2018). *Intrusion Detection with Comparative Analysis of Supervised Learning Techniques and Fisher Score Feature Selection Algorithm*. [online] Available from: https://www.researchgate.net/publication/327700381 [Accessed 29 May 2020].

Baig, Z., Fu, X., Ge, M., Robles-Kelly, A., Syed, N., and Teo, G. (2019). *Deep Learning-based Intrusion Detection for IoT Networks*. [online] Available from: https://ieeexplore.ieee.org/document/8952154 [Accessed 27 July 2020].

Bermudez, L. (2017). *Overview of Neural Networks*. [online] Available from: https://medium.com/machinevision/overview-of-neural-networks-b86ce02ea3d1 [Accessed 27 July 2020].

Borah, S. and Panigrahi, R. (2018). *A detailed analysis of CICIDS2017 dataset for designing Intrusion Detection Systems*. In: *International Journal of Engineering & Technology*. [online] Science Publishing Corporation, pp.479-482. Available from: https://www.researchgate.net/publication/329045441 [Accessed 29 May 2020].

Brownlee, J. (2020). *How to Calculate Precision, Recall, and F-Measure for Imbalanced Classification*. [online] Available from: https://machinelearningmastery.com/precision-recall-and-f-measure-for-imbalanced-classification/ [Accessed 6 August 2020].

Capgemini Research Institute. (2019). *Reinventing Cybersecurity with Artificial Intelligence*. [online] Available from: https://www.capgemini.com/wpcontent/uploads/2019/07/AI-inCybersecurity_Report_20190711_V06.pdf [Accessed 29 May 2020].

D'hooge, L., De Turck, F., Volckaert, B., and Wauters, T. (2019). *In-depth Comparative Evaluation of Supervised Machine Learning Approaches for Detection of Cybersecurity Threats*. [online] Available from: https://www.scitepress.org/Link.aspx?doi=10.5220/0007724801250136 [Accessed 29 May 2020].

Dogdu, E. and Faker, O. (2019). *Intrusion Detection Using Big Data and Deep Learning Techniques*. [online] Available from: https://doi.org/10.1145/3299815.3314439 [Accessed 29 May 2020].

Ferraga, M., Janicke, H., Maglaras, L., and Moschoyiannis, S. (2019). *Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study*. In: Ho, A. (ed).

Journal of Information Security and Applications. [online] Elsevier. Available from: https://doi.org/10.1016/j.jisa.2019.102419 [Accessed 1 July 2020].

Géron, A. (2019). *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow*. Sebastopol: O'Reilly Media, Inc.

Ghorbani, A., Lashkari, A., and Sharafaldin, I. (2018). *Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization*. [online] Available from: https://www.scitepress.org/Link.aspx?doi=10.5220/0006639801080116 [Accessed 29 May 2020].

Hassan, W. and Noor, M. (2018). *Current research on Internet of Things (IoT) security: A survey*. In: Iera, A. and Melodia, T. (ed). *Computer Networks*. [online] Elsevier, pp.283-294. Available from: https://doi.org/10.1016/j.comnet.2018.11.025 [Accessed 24 July 2020].

Koroniotis, N., Moustafa, N., Sitnikova, E., Turnbull, B. (2018). *Towards the Development of Realistic Botnet Dataset in the Internet of Things for Network Forensic Analytics: Bot-IoT Dataset*. [online] Available from: https://arxiv.org/abs/1811.00701 [Accessed 4 July 2020].

Liao, H., Lin, C., Lin, Y., and Tung, K. (2012). *Intrusion detection system: A comprehensive review*. In: Atiquzzaman, M. (ed). *Journal of Network and Computer Applications*. [online] Elsevier, pp.16-24. Available from: https://doi.org/10.1016/j.jnca.2012.09.004 [Accessed 24 July 2020].

Mahmoud, Q. and Ullah, I. (2020). *A Two-Level Flow-Based Anomalous Activity Detection System for IoT Networks*. [online] Available from: https://www.mdpi.com/2079-9292/9/3/530 [Accessed 27 July 2020].

Morde, V. (2019). *XGBoost Algorithm: Long May She Reign!* [online] Available from: https://towardsdatascience.com/https-medium-com-vishalmorde-xgboost-algorithm-long-she-may-rein-edd9f99be63d [Accessed 27 July 2020].

SAS Institute. (2020). *Machine Learning: What it is and why it matters*. [online] Available from: https://www.sas.com/en_gb/insights/analytics/machine-learning.html [Accessed 3 June 2020].

Taylor, H. (2020). *What Are Cyber Threats and What to Do About Them*. [online] Available from: https://preyproject.com/blog/en/what-are-cyber-threats-how-they-affect-you-what-to-do-about-them/ [Accessed 3 June 2020].

The Institute of Risk Management (IRM). (2019). *10 staggering cybersecurity statistics for 2019*. [online] Available from: https://www.irmsecurity.com/resources/10-staggering-cybersecurity-statistics-for-2019/ [Accessed 3 June 2020].

APPENDICES

APPENDIX A

Model	Dataset	Binary Configuration	Multiclass Configuration
XGBoost	CICIDS2017	• n_estimators=125 • learning_rate=0.3 • max_depth=10 • gamma=0.5 • subsample=1 • colsample_bytree=1 • objective='binary:logistic' • nthread=4 • seed=42	• n_estimators=125 • learning_rate=0.3 • max_depth=10 • gamma=0.5 • subsample=1 • colsample_bytree=1 • objective='multi:softmax' • num_class=7 • nthread=4 • seed=42
	BoT-IoT	• n_estimators=10 • max_depth=1 • min_child_weight=20 • gamma=20 • subsample=0.5 • colsample_bytree=0.5 • objective='binary:logistic' • nthread=4 • seed=42	• n_estimators=10 • max_depth=1 • min_child_weight=20 • gamma=20 • subsample=0.5 • colsample_bytree=0.5 • objective='multi:softmax' • num_class=4 • nthread=4 • seed=42
Neural Network	CICIDS2017	• Input layer ○ 70 neurons • Hidden layer 1 ○ 256 neurons ○ activation='relu' • Hidden layer 2 ○ 128 neurons ○ activation='relu' • Hidden layer 3 ○ 64 neurons ○ activation='relu' • Output layer ○ 1 neuron	• Input layer ○ 70 neurons • Hidden layer 1 ○ 256 neurons ○ activation='relu' • Hidden layer 2 ○ 128 neurons ○ activation='relu' • Hidden layer 3 ○ 64 neurons ○ activation='relu' • Output layer ○ 7 neurons

		○ activation=’sigmoid’ • loss=’binary_crossentropy’ • optimizer=’adam’ • epochs=10 • batch_size=512	○ activation=’softmax’ • loss=’categorical_crossentropy’ • optimizer=’adam’ • epochs=10 • batch_size=512
	BoT-IoT	• Input layer ○ 177 neurons • Hidden layer ○ 16 neurons ○ activation=’relu’ • Dropout layer ○ rate=0.5 • Output layer ○ 1 neuron ○ activation=’sigmoid’ • loss=’binary_crossentropy’ • optimizer=’adam’ • epochs=10 • batch_size=256	• Input layer ○ 177 neurons • Hidden layer ○ 16 neurons ○ activation=’relu’ • Dropout layer ○ rate=0.5 • Output layer ○ 4 neurons ○ activation=’softmax’ • loss=’categorical_crossentropy’ • optimizer=’adam’ • epochs=10 • batch_size=256